**Voici un modèle conceptuel et relationnel basé sur notre description fournie :**

**Modèle Conceptuel :**

- Ville (ville\_id, nom, population, superficie)

- Structure (structure\_id, ville\_id, type\_structure, capacite)

- Stationnement (stationnement\_id, ville\_id, structure\_id, statut)

- Utilisateurs (utilisateur\_id, nom, prenom, adresse, email)

- Véhicules (vehicule\_id, utilisateur\_id, marque, dimensions, constructeur)

- Déplacements (deplacement\_id, vehicule\_id, date\_depart, date\_arrivee, distance)

- Pool de demande (demande\_id, utilisateur\_id, date\_demande, structure\_id)

- Coordonnées (coordonnees\_id, structure\_id, latitude, longitude)

- Historique (historique\_id, table, action, date\_action)

**Modèle Relationnel :**

- Ville (ville\_id [PK], nom, population, superficie)

- Structure (structure\_id [PK], ville\_id [FK], type\_structure, capacite)

- Stationnement (stationnement\_id [PK], ville\_id [FK], structure\_id [FK], statut)

- Utilisateurs (utilisateur\_id [PK], nom, prenom, adresse, email)

- Véhicules (vehicule\_id [PK], utilisateur\_id [FK], marque, dimensions, constructeur)

- Déplacements (deplacement\_id [PK], vehicule\_id [FK], date\_depart, date\_arrivee, distance)

- Pool de demande (demande\_id [PK], utilisateur\_id [FK], date\_demande, structure\_id [FK])

- Coordonnées (coordonnees\_id [PK], structure\_id [FK], latitude, longitude)

- Historique (historique\_id [PK], table, action, date\_action)

Explications :

- **Le modèle conceptuel** représente les entités et leurs relations sans tenir compte des contraintes de clés primaires (PK) ou étrangères (FK).

**- Le modèle relationnel** traduit le modèle conceptuel en tables avec des clés primaires et étrangères.

- Chaque table est décrite avec ses attributs correspondants. Les identifiants (ID) sont utilisés comme clés primaires.

- Les relations entre les tables sont établies à l'aide des clés primaires et étrangères.

**Différentes associations possibles dans le model conceptuel :**

Voici les associations possibles dans le modèle conceptuel donné :

1. Ville - Structure :

- Une ville peut avoir plusieurs structures.

- Une structure est associée à une seule ville.

2. Ville - Stationnement :

- Une ville peut avoir plusieurs stationnements.

- Un stationnement est associé à une seule ville.

3. Structure - Stationnement :

- Une structure peut avoir plusieurs stationnements.

- Un stationnement est associé à une seule structure.

4. Utilisateurs - Véhicules :

- Un utilisateur peut posséder plusieurs véhicules.

- Un véhicule appartient à un seul utilisateur.

5. Véhicules - Déplacements :

- Un véhicule peut être utilisé pour plusieurs déplacements.

- Un déplacement est effectué par un seul véhicule.

6. Utilisateurs - Pool de demande :

- Un utilisateur peut faire partie de plusieurs pools de demande.

- Un pool de demande est associé à un seul utilisateur.

7. Utilisateurs - Historique :

- Un utilisateur peut avoir plusieurs entrées dans l'historique.

- Une entrée dans l'historique est liée à un seul utilisateur.

8. Structure - Coordonnées :

- Une structure peut avoir plusieurs enregistrements de coordonnées.

- Des coordonnées sont associées à une seule structure.

Ces associations permettent de représenter les relations entre les entités du modèle conceptuel.

**Implantation du modèle relationnel dans le SGBD :**

CREATE TABLE Ville (

ville\_id INT PRIMARY KEY,

nom VARCHAR(30),

population INT,

superficie FLOAT

);

CREATE TABLE Structure (

structure\_id INT PRIMARY KEY,

ville\_id INT,

type\_structure VARCHAR(30),

capacite INT,

FOREIGN KEY (ville\_id) REFERENCES Ville(ville\_id)

);

CREATE TABLE Stationnement (

stationnement\_id INT PRIMARY KEY,

ville\_id INT,

structure\_id INT,

statut VARCHAR(30),

FOREIGN KEY (ville\_id) REFERENCES Ville(ville\_id),

FOREIGN KEY (structure\_id) REFERENCES Structure(structure\_id)

);

CREATE TABLE Utilisateurs (

utilisateur\_id INT PRIMARY KEY,

nom VARCHAR(30),

prenom VARCHAR(30),

adresse VARCHAR(30),

email VARCHAR(30)

);

CREATE TABLE Vehicules (

vehicule\_id INT PRIMARY KEY,

utilisateur\_id INT,

marque VARCHAR(30),

dimensions VARCHAR(30),

constructeur VARCHAR(30),

FOREIGN KEY (utilisateur\_id) REFERENCES Utilisateurs(utilisateur\_id)

);

CREATE TABLE Deplacements (

deplacement\_id INT PRIMARY KEY,

vehicule\_id INT,

date\_depart DATE,

date\_arrivee DATE,

distance FLOAT,

FOREIGN KEY (vehicule\_id) REFERENCES Vehicules(vehicule\_id)

);

CREATE TABLE Pool\_demande (

demande\_id INT PRIMARY KEY,

utilisateur\_id INT,

date\_demande DATE,

structure\_id INT,

FOREIGN KEY (utilisateur\_id) REFERENCES Utilisateurs(utilisateur\_id),

FOREIGN KEY (structure\_id) REFERENCES Structure(structure\_id)

);

CREATE TABLE Coordonnees (

coordonnees\_id INT PRIMARY KEY,

structure\_id INT,

latitude FLOAT,

longitude FLOAT,

FOREIGN KEY (structure\_id) REFERENCES Structure(structure\_id)

);

CREATE TABLE Historique (

historique\_id INT PRIMARY KEY,

table\_name VARCHAR(30),

action VARCHAR(30),

date\_action DATE

);

**Création du code Arduino avec le SGBD :**

Voici un exemple de code Arduino qui utilise un SGBD pour communiquer avec une base de données correspondant au schéma :

```cpp

#include <SPI.h>

#include <Ethernet.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

// Définir les informations de connexion à la base de données

IPAddress server(192, 168, 1, 100);

char user[] = "utilisateur";

char password[] = "mot\_de\_passe";

char db[] = "nom\_base\_de\_donnees";

EthernetClient client;

MySQL\_Connection conn(&client);

void setup() {

Serial.begin(9600);

// Connexion à la base de données

if (conn.connect(server, 3306, user, password)) {

Serial.println("Connexion à la base de données réussie !");

} else {

Serial.println("Échec de la connexion à la base de données.");

while (1);

}

}

void loop() {

// Exemple d'exécution d'une requête SELECT

MySQL\_Cursor \*cur\_mem = new MySQL\_Cursor(&conn);

char query\_select[] = "SELECT \* FROM Ville";

cur\_mem->execute(query\_select);

MySQL\_Row row;

do {

row = cur\_mem->get\_next\_row();

if (row.length() > 0) {

int ville\_id = atoi(row.get\_field(0));

char\* nom = row.get\_field(1);

int population = atoi(row.get\_field(2));

float superficie = atof(row.get\_field(3));

// Faire quelque chose avec les résultats (par exemple, les afficher)

Serial.print("Ville ID : ");

Serial.println(ville\_id);

Serial.print("Nom : ");

Serial.println(nom);

Serial.print("Population : ");

Serial.println(population);

Serial.print("Superficie : ");

Serial.println(superficie);

}

} while (row.length() > 0);

delete cur\_mem;

delay(5000); // Attendre 5 secondes avant de réexécuter la requête

}

```

**Assurez-vous d'installer les bibliothèques Ethernet et MySQL Connector/Arduino dans votre environnement Arduino IDE.** N'oubliez pas de remplacer les informations de connexion à la base de données (`server`, `user`, `password`, `db`) par les valeurs appropriées.

Ce code établit une connexion à la base de données, exécute une requête SELECT pour récupérer les données de la table "Ville" et affiche les résultats sur le moniteur série. Vous pouvez adapter ce code pour interagir avec les autres tables de votre schéma de base de données en utilisant des requêtes SQL appropriées.

**Pour récupérer les informations de la base de données (SGBD) à partir d'un code Arduino, vous avez besoin d'établir une connexion avec la base de données et exécuter des requêtes SQL pour récupérer les données souhaitées.** Cependant, il convient de noter que le langage Arduino n'a pas nativement de support pour interagir directement avec un SGBD. Vous aurez donc besoin d'une interface supplémentaire, telle qu'un module WiFi ou Ethernet, pour établir la connexion avec la base de données.

Voici un exemple de code Arduino qui utilise une connexion WiFi pour récupérer les informations de la base de données MySQL :

```cpp

#include <WiFi.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

// Paramètres de connexion WiFi

const char\* ssid = "nom\_du\_reseau";

const char\* password = "mot\_de\_passe";

// Paramètres de connexion à la base de données MySQL

IPAddress server\_addr(xxx, xxx, xxx, xxx); // Adresse IP du serveur MySQL

int server\_port = 3306; // Port du serveur MySQL

const char\* user = "utilisateur";

const char\* password\_db = "mot\_de\_passe\_db";

const char\* database = "nom\_bdd";

// Objet pour la connexion MySQL

WiFiClient client;

MySQL\_Connection conn((Client \*)&client);

void setup() {

Serial.begin(115200);

// Connexion au réseau WiFi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connexion au WiFi en cours...");

}

// Connexion à la base de données MySQL

Serial.print("Connexion à la base de données MySQL : ");

if (conn.connect(server\_addr, server\_port, user, password\_db)) {

Serial.println("OK");

} else {

Serial.println("Échec de la connexion");

}

}

void loop() {

// Exécuter une requête SELECT pour récupérer les informations des tables

MySQL\_Cursor\* cursor = new MySQL\_Cursor(&conn);

char query[128];

// Exemple : récupérer toutes les données de la table Ville

sprintf(query, "SELECT \* FROM Ville");

cursor->execute(query);

MySQL\_Row row;

do {

row = cursor->get\_next\_row();

if (row.length() > 0) {

for (int i = 0; i < row.length(); ++i) {

Serial.print(row[i]);

Serial.print("\t");

}

Serial.println();

}

} while (row.length() > 0);

delete cursor;

delay(5000); // Attendre 5 secondes avant d'exécuter la requête suivante

}

```

**Assurez-vous d'installer les bibliothèques nécessaires dans votre environnement Arduino IDE. Dans cet exemple, j'ai utilisé les bibliothèques "WiFi", "MySQL\_Connection" et "MySQL\_Cursor".** Vous pouvez les installer en allant dans le menu "Sketch -> Include Library -> Manage Libraries".

N'oubliez pas de remplacer les paramètres tels que le nom du réseau WiFi, le mot de passe, l'adresse IP du serveur MySQL, les identifiants de connexion à la base de données et le nom de la base de données par vos propres valeurs.

Cet exemple illustre la récupération des données de la table "Ville". Vous pouvez adapter le code en fonction de vos besoins pour récupérer les informations d'autres tables de la base de données. N'oubliez pas d'établir les connexions et d'exécuter les requêtes SQL appropriées pour chaque table que vous souhaitez interroger.