NAJDEK Thomas
MACKOWIAK Carole
SAINT-MACHIN Nathan
LEGRAND Pierre-Yves

Groupe C

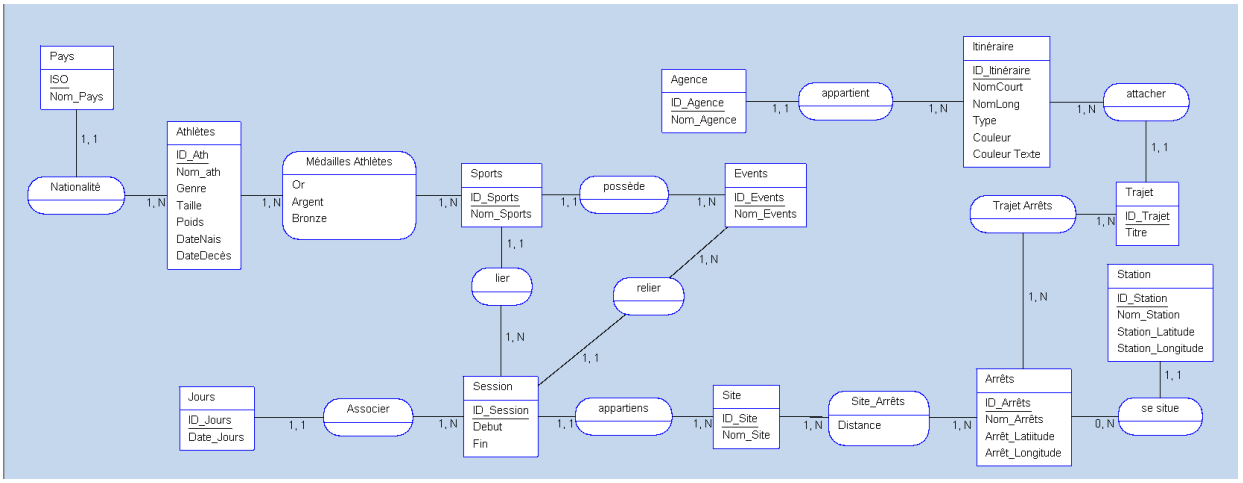
Rapport SAE BD

Semaine `SELECT semaine FROM calendrier NATURAL JOIN devoirs WHERE devoirs.nom LIKE “Rapport SAE”`

2022-2023

SOMMAIRE

I/ Modèle Entité-Association



Note: Le modèle EA a été altéré depuis le dernier rapport.

I/ Données de data.gouv.fr

Une partie des données ont été récupérées via le site *data.gouv.fr*, une plateforme créée par le gouvernement français dans le but est de fournir un accès libre et gratuit aux données publiques sous forme numérique. Les données publiques comprennent des informations provenant de différents domaines tels que l'économie, l'environnement, la santé, l'éducation, etc..

Depuis ce site ont été récupérés les éléments de l'entité Pays, qui contient donc la liste des pays participants aux Jeux Olympiques. Ces derniers sont représentés par leur code ISO (la norme internationale pour les noms des pays), et contiennent le nom long du pays en anglais. Les informations sont téléchargeables dans le format CSV, et peuvent être trouvées sur ce lien : https://www.data.gouv.fr/fr/datasets/winter-olympics-medals/. Elles sont essentielles pour mettre une identification claire des pays dans le cadre des Jeux Olympiques.

De plus, l'entité Sites représente les différents sites où les sessions d'entraînement ou de compétition peuvent avoir lieu. Chaque site est caractérisé par son identifiant unique, son nom et ses coordonnées géographiques. Ces dernières ont été récupérées dans un fichier JSON également fourni sur data.gouv.fr, et téléchargeable ici : https://www.data.gouv.fr/fr/datasets/paris-2024-sites-olympiques-et-paralympiques-franciliens/

D'autres sites ont été récupérés via le calendrier provisoire disponible sur paris2024.org ; ces derniers n'étant pas présents dans le fichier JSON, nous avons dû entrer leurs coordonnées géographiques manuellement.

Pour insérer toutes ces informations dans la base de données, nous avons utilisé des scripts Python à l'aide des modules json et csv pour lire et parser les fichiers, ainsi que PyMySQL pour se connecter à la base de données.

PyMySQL est une bibliothèque Python qui permet de se connecter à une base de données MySQL et d'exécuter des commandes SQL qui permettent de réaliser des opérations telles que l'insertion, la mise à jour et la récupération de données. L'utilisation de cette librairie, nous permet de traiter automatiquement les données et d'effectuer des opérations à grande échelle.

II/ Données de olympedia.org

Olympedia.org est un site internet qui contient des informations historiques sur les Jeux Olympiques ; on peut par exemple y trouver des informations sur les sports et les résultats des éditions précédentes, ainsi que sur les athlètes et les médailles obtenues par ces derniers. Nous avons donc utilisé ce site pour obtenir les informations suivantes.

Les données pour 2024 n'étant pas disponibles, l'entité Athlètes représente les informations relatives aux athlètes qui ont participé aux éditions précédentes. Chaque athlète est représenté par un identifiant unique (récupéré depuis Olympedia), et possède un nom, un genre, une nationalité, et si les informations suivantes sont disponibles, une taille, un poids, une date de naissance ou de décès.

La relation *Médaille athlètes* représente les médailles remportées par les athlètes dans les différents sports. Elle relie les entités Athlètes et Sports en utilisant les clé étrangère ID_Ath et ID_Sport. En plus des clés étrangères, la relation *Médaille athlètes* comprend trois attributs pour enregistrer le nombre de médailles d'or, d'argent et de bronze remportées pour l'athlète concerné.

Pour récolter ces informations depuis Olympedia, nous avons utilisé différents modules Python.

- Le principal est *requests* : ce module permet de faire des requêtes HTTP de façon simplifiée, pour pouvoir récupérer le code HTML des pages de Olympedia.
- Le module *re* (inclus dans Python) a été utilisé pour extraire de façon rapide les informations depuis le code HTML grâce à des expressions régulières (Regex).
- *PyMySQL* est de nouveau utilisé pour interagir avec la base de données et y insérer les informations.

Notre programme Python définit une fonction '**get**' pour récupérer le contenu des pages Web et les mettre en cache localement (afin d'éviter de spammer le site Olympedia à chaque fois que le programme est démarré).

III/ Calendrier de paris2024.org

Le site paris2024.org fournit un fichier contenant un calendrier provisoire et détaillé pour les Jeux Olympiques et Paralympiques 2024. On y trouve chaque sport associé à son identifiant à 3 lettres, ainsi que toutes les sessions pour chaque sport, auxquelles sont associés des sites olympiques et différents événements.

- L'entité Sport représente les différents sports qui se déroulent lors des JO 2024, qui sont caractérisés par leur code à 3 lettres ainsi que par leur nom (en anglais dans la base de données).

- L'entité Session représente les différentes sessions pour chaque sport : elles possèdent un site, un numéro de jour, ainsi que des horaires de début et de fin. Par exemple, pour le sport Archery (ARC), la Session 1 se déroule le 2024-07-25 de 9:30 à 12:30 aux Invalides.

- L'entité Events représente les différents évènements qui se déroulent lors d'une session. Par exemple, lors de la session 1 de ARC se déroule le "Women's Individual Ranking Round".

- L'entité Jours représente la date associée aux numéros de jours.

Pour pouvoir lire ce fichier PDF, nous avons d'abord utilisé le module *pdfminer*, qui contient un script *pdf2txt* qui permet de convertir un fichier PDF dans un format plus facilement exploitable ; dans notre cas, il s'agit d'un fichier XML. Ce fichier contient donc la position de chaque caractère, qui se situent dans des lignes, qui se situent dans des blocs.

Ce fichier XML a été parsé avec la librairie standard *xml ElementTree*. Nous avons donc pu lire le fichier PDF ligne par ligne, ce qui nous a permis d'y récupérer les informations dont nous avons besoin.

Nous avons créé trois classes pour stocker les informations sur les sports, les jours et les sessions : '**Sport**', '**Day**' et '**Session**'.

Pour traiter ces informations contenues dans ce fichier, nous avons initialisé trois variables : '**currentSport**', '**currentDay**' et '**currentSession**' pour garder une trace du sport, du jour et de la session en cours de traitement.

Nous avons également créé une fonction '**saveMeta**' pour enregistrer les métadonnées. Ces informations sont stockées dans une liste globale appelée '**meta'**, et la fonction ajoute ces informations à l'attribut d'événements de la session en cours.

Enfin, nous avons utilisé une fonction '**handle Line**' pour analyser chaque ligne du texte. Cette fonction vérifie d'abord si la ligne doit être ignorée et s'arrête immédiatement si c'est le cas.

Par exemple :
Si la ligne contient le mot “Session”, la fonction tente d’extraire des informations sur la session à l’aide d’une expression régulière.
Si la correspondance est trouvée, la fonction crée un nouvel objet Session et l'attribue à la session en cours. Si aucune correspondance n’est trouvée, une exception est levée.

III/ Données de Île de France Mobilités

L’entité Itinéraire stocke les informations de base sur un itinéraire, comme son identifiant unique, le nom court et long de l’itinéraire, l’agence qui l’exploite, le type de véhicule utilisé et les couleurs utilisées pour l’affichage de l’itinéraire.

L’entité trajet représente les trajets effectués par chaque itinéraire.Chaque trajet a son propre identifiant unique et un nom descriptif.
La relation entre trajet et la table itinéraire est une relation de clé étrangère, car chaque trajet est associé à un itinéraire spécifique, identifiable par son ID_Itinéraire.

L’entité arrêts stocke les informations sur les différents arrêts desservis par les itinéraires. Chaque arrêt à son propre identifiant unique, un nom descriptif , ainsi que des informations de géolocalisation représentées par les coordonnées de latitude et longitude.

Trajet Arrêt établit une relation entre les trajets et les arrêts desservis. Elle stocke la position de chaque arrêt dans le trajet, en utilisant un numéro de position, ainsi que l’identifiant unique du trajet et de l’arrêt associé.

La relation Site_Arrêts représente les sites desservis par les itinéraires, qui peuvent être des arrêts, mais aussi des lieux d’intérêts. Elle stocke la distance entre chaque site et chaque arrêt desservi par l’itinéraire, ainsi que les identifiants uniques du site et de l’arrêt associé.

Pour remplir nos tables restantes, on importe différents modules csv qu’on a utiliser ci dessus, geopy.distance , PyMySQL, sys , os , re et Décimal.

Nous avons créé plusieurs classes Stop , Station , Route , Trip , StopTime et Site.

La classe Station illustre une station sur une ligne de transport en commun. Elle a des propriétés similaires à celles de la classe Stop, sauf qu’elle n’a pas de parent de Station.

La classe Stop représente un arrêt sur une ligne de transport. Elle a des propriétés telles que son identifiant, le nom, la longitude et la latitude de l’arrêt, et le parent de la Station.

La classe route représente une ligne de transport en commun. Elle est caractérisée par son identifiant de la ligne, l’identifiant de l’agence, le nom court et long de la ligne, le type de transport, couleur et couleur du texte de la ligne.

La classe Trip est un trajet sur une ligne de transport en commun . Elle possède l’identifiant de la ligne, l'identifiant du service , l’identifiant du trajet, la destination du trajet et l’identifiant de direction.

La classe Stop Time représente les heures de passage d'un trajet à un arrêt spécifique. Elle a l'identifiant du trajet , l'heure d'arrivée , l'heure de départ , l'identifiant et la séquence de l'arrêt.

La classe Site représente comme son nom l'indique un site olympique et elle possède les mêmes caractéristiques que la table Site.

Notre script nous permet de lire un fichier stops.txt qui contient les informations sur les arrêts et les stations de transport en commun. Il crée des objets Stop pour les arrêts et des objets Stations pour les stations. Nous ignorons tous les autres types de lieux.

Les objets Stop et Stations sont stockés dans le dictionnaire stops et stations respectivement. Ensuite on lit les sites olympiques déjà présents dans notre base de données et on les stocke dans un dictionnaire Sites.

Juste après on calcule les arrêts de transports en commun les plus proches de chaque site. C'est pour cela qu'on a décidé de stocker les données de géolocalisation de ces sites et les arrêts. Et enfin nous faisons en sorte de générer un plan de transport pour chaque jour des JO,en déterminant les itinéraires les plus optimaux et le plan de transport final prend en compte les horaires des compétitions et des événements afin de minimiser le temps de trajet pour les athlètes, les spectateurs et les officiels.

II/ Modèle Relationnel

Athlètes (ID_Ath , Nom_Athlète , Genre, Taille , Poids, Date_Naissance, Date_Dècès)

- Athlètes :
    - ID_Ath : Clé primaire (int)
    - Nom_Athlète : Chaîne de caractère (Varchar (255))
    - Genre : Enum ('F', 'M')
    - Taille : Nombre entier (int)
    - Poids : Nombre entier (int)
    - Date_Naissance : Date (Date)
    - Date_Dècès : Date (Date)

Sports (ID_Sports , Nom_Sports)

- Sports :
    - ID_Sports : Clé Primaire (char (3))
    - Nom_Sports : Chaine de Caractère (Varchar (255))

Pays (ISO , Nom_Pays)

- Pays :
    - ISO : Clé Primaire (char (3))
    - Nom_Pays : Chaîne du caractère (Varchar (255))

Jours (ID Jours , Date)

- Jours :
    - ID_Jours : Clé primaire (int)
    - Date_Jours : Date (Date)

Events(ID_Events, Nom_Events , ID_Sport*, ID_Session*)

- Events :
    - ID_Events : Clé primaire (Int)
    - Nom_Events : Chaîne de caractère (Varchar (255))
    - ID_Sport* : Clé Étrangère
    - ID_Session* : Clé Étrangère

Arrêts (ID_Arrêts , Nom_ Arrêts, Arrêt_Latitude , Arrêt_Longitude)

- Arrêts :
    - ID_Arrêts : Clé Primaire (int)
    - Nom_ Arrêts : Chaîne de caractère (varchar (255))
    - Arrêt_Latitude : Nombre décimal (Decimal)
    - Arrêt_Longitude : Nombre décimal (Decimal)

Trajet (ID_Trajet , Nom_Trajet, ID Itinéraire*)

- Trajet :
    - ID_Trajet : Clé primaire (int)
    - Nom_Trajet : Chaine de caractère (varchar (255))
    - ID_Itinéraire* : Clé étrangère (Itinéraire)

Itinéraire (ID_Itinéraire , Nom_Court, Nom_Long , Type , Couleur , Couleur Texte)

- Itinéraire :
    - ID_Itinéraire : Clé Primaire (varchar(64))
    - Nom_Court : Chaîne de caractère (varchar(50))
    - Nom_Long : Chaîne de caractère (varchar(255))
    - Type : Nombre entier (Int)
    - Couleur : Nombre entier (Int)
    - Couleur Texte : Nombre entier (Int)

Station (ID_Station, Nom_Station, Station_Latitude, Station_Longitude)

- Station :
    - ID_Station : Clé Primaire (Varchar (64))
    - Nom_Station : Chaîne de caractère (Varchar (255))
    - Station_Latitude : Nombre décimal (Décimal)
    - Station_Longitude : Nombre décimal (Décimal)

Agence(ID_Agence, Nom_Agence)

- Agence :
    - ID_Agence : Clé primaire (varchar(64))
    - Nom_Agence : Chaîne de caractère (varchar(255))

Sites(Id_Site, Nom_Site,Site_Latitude ,Site_Longitude)

- Sites :
    - ID_Sites : Clé primaire (Int)
    - Nom_Site : Chaîne de caractère (varchar(255))
    - Site_Latitude : Nombre décimal (Décimal)
    - Site_Longitude : Nombre décimal (Décimal)

Médaille Athlètes (Or , Argent , Bronze, ID_Ath* , ID_Sport*)

- Médaille Athlètes :
    - ID_Ath* : Clé Étrangère (Athlètes)
    - ID_Sport* : Clé Étrangère (Sports)
    - Or : Nombre entier (int)
    - Argent : Nombre entier (int)
    - Bronze : Nombre entier (int)

Session (ID_Session, Début , Fin ,ID_Jours*, ID_Site* , ID_Sport*)

- Session :
    - ID_Session : Clé Primaire (Int)
    - Début : Temps (Time)
    - Fin : Temps (Time)
    - ID_Jours* : Clé Étrangère (Jours)
    - ID_Sites* : Clé Étrangère (Sites)
    - ID_Sports* : Clé Étrangère (Sports)

Site_Arrêts (Distance , ID_Site* , ID_Arrêts*)

- Site_Arrêts :
    - Distance : Nombre entier (Int)
    - Id_Sites* : Clé étrangère (Sites)
    - ID_Arrêts* : Clé étrangère (Arrêts)

Trajet_Arrêts (Position , ID_Trajet*,ID_Arrêts*)

- Trajet_Arrêts :
    - Position : Nombre entier (Int)
    - ID_Trajet* : Clé étrangère (Trajet)
    - ID_Arrêts* : Clé étrangère (Arrêts)

Nationalité(ISO* , ID_Ath*)

- Nationalité :
    - ISO : Clé étrangère (Pays)
    - ID_Ath : Clé étrangère (Athlète)
    -

Appartient (ID_Agence* , ID_Itinéraire*)

- Appartient :
    - ID_Agence : Clé étrangère (Agence)
    - ID_Itinéraire : Clé étrangère (Itinéraire)

Se situe (ID_Arrêts*, ID_Station*)

- Se situe :
    - ID_Arrêts : Clé étrangère (Arrêts)
    - ID_Station : Clé étrangère (Station)

III/ Consultations des données

Pour consulter nos données, nous avons opté pour un site web pour pouvoir accéder le plus facilement aux informations de notre base de données.
Ainsi, nous avons utilisé une application web Flask qui nous permet de fournir toutes les informations que nous avons besoins comme les sites, arrêts desservis par les transports en commun que cela soit lors des Jeux Olympiques et Paralympiques de 2024.

Nous avons importé différents modules dans notre script, Flask , render_template,request qui nous permet de créer une application web Flask, de générer des pages HTML à partir de modèles, et pour traiter les requêtes du client.
PyMySQL et re sont des modules que nous avons déjà identifiés dans la première partie.

Ensuite, l'application Flask est initialisée avec le nom de l'application et le chemin de l'URL statique. Le paramètre TEMPLATES_AUTO_RELOAD est défini à True pour permettre à Flask de recharger automatiquement les modèles de page chaque fois qu'ils sont modifiés.

Nous avons implémenté une fonction process_host qui est un décorateur de contexte, qui injecte des variables globales dans nos templates ; cette dernière vérifie le domaine auquel accède l'utilisateur. Si ce dernier contient le mot "para", alors l'utilisateur souhaite très certainement accéder au site des Paralympiques 2024. Dans le cas contraire, il s'agit du site des Olympiques 2024.

Nous avons la fonction get_picto_itinéraire(agence, type) qui nous permet d'obtenir le chemin vers une image représentant un itinéraire de transport en commun.

Ensuite nous avons la fonction connect_sql() permet de nous connecter à notre base de données MySQL.

Enfin, nous avons 3 fonctions d'accueil , sites et sites(SiteID) qui sont des fonctions de vue Flask qui nous renvoient le contenu HTML pour les différentes pages du site.
Comme son nom l'indique, accueil renvoie simplement le modèle pour la page d'accueil.
Site() renvoie une liste des sites des Jeux Olympiques ou Paralympiques avec leurs coordonnées géographiques.
Site(SiteID) renvoie une page avec des informations détaillées sur un site particulier, y compris les sports proposés, les arrêts desservis et les itinéraires de transport en commun associés.

IV/ Conclusion

Cette SAÉ nous a permis de créer une base de données assez complète contenant une grande variété d'informations pertinentes pour les Jeux Olympiques et Paralympiques.
Nous avons aussi découvert Flask et développé un site Web ou l'on peut consulter ces données de manière efficace.
Cela nous permet de mieux comprendre l'ampleur et la complexité des Jeux Olympiques et Paralympiques, ainsi que les défis techniques liés à la création d'une base de données et d'un site web fonctionnels et accessibles.