# Examen Machine - Bases de Données

CS54 : Computer Science 54
Première année

15 Octobre 2022
Durée : 3 heures

## Propos liminaires

Vous disposez de 3 heures pour réaliser tout ou partie de l'examen. Prenez le temps de lire les questions, de les comprendre, de réfléchir à des solutions. Il existe de multiples manières de résoudre les problèmes posés.

Assurez-vous de bien déposer vos productions sur Arche (1 dossier compressé contenant plusieurs fichiers de réponse (1 fichier .sql par question)), les évaluations étant automatiques. Ne copiez pas, ne trichez pas. Les conséquences (conseil de discipline, exclusion, …) sont trop lourdes pour s'y risquer.

## Connexion au serveur de base de données Postgres SQL :

Veuillez lancer **pgAdmin** et créez une instance du serveur avec l'option "*Create Server* ".
Choisissez un nom de connexion et utilisez *localhost* comme adresse du serveur (port *5432*, nom d'utilisateur : *postgres* et mot de passe : *votre mot de passe (défini lors de l'installation de postgressql)*).
Pour interroger le schéma de la base de données, il suffit de sélectionner le sous menu *postgres* du menu *Databases* de votre serveur et puis de lancer l'outil "*Query Tool*" (Menu *Tools* de l'interface principale de ***pgAdmin***).
Saisissez vos requêtes dans la fenêtre *Query editor* de l'outil *Query tool* !
Vous pouvez aussi utiliser Postgres SQL en ligne de commande.

**Supports de cours :** Les supports de cours sont disponibles sur Arche.
**Documentation de PostgresSQL :** https://www.postgresql.org/docs/
**Tutorial avec des éléments de syntaxe PL/PGSQL :**
https://www.postgresqltutorial.com/postgresql-plpgsql/

# Sujet d'examen et modalités de rendu

- Le sujet d'examen est disponible sur Arche.
- Vous sauvegarderez au fur et à mesure vos **requêtes** dans des fichiers sql (.sql). Les noms des fichiers doivent respecter le format suivant : **Q_numéro-de-la-question.sql (par exemple : Q_1.sql).** L'ensemble des fichiers

"**.sql"** doivent être placés dans un dossier compressé qui doit respecter le format suivant : **NOM_PENOM_TP_BD.zip.**

# C'est parti !

Des clubs de ski souhaitent informatiser la gestion des compétitions. Pour cela, ils utilisent une base de données dans laquelle les relations suivantes sont définies (**clé primaires** en gras et *clés étrangères* précédées par #) :

**Station** (**nom_station** , alt_station , pays_station , capacite)
*Cette relation donne le nom (ex. Tignes, Bormio…), l'altitude (en mètres), le pays (ex. France, Italie…) et la capacité d'accueil d'une station de ski (nombre de lits)*

**Skieur** (**no_skieur** , nom_skieur , prenom , *#nom_station*)
*Cette relation décrit un skieur : le numéro, le nom, le prénom d'un skieur ainsi que le nom de sa station de rattachement ou d'origine*

**Competition** (**no_comp**, date_comp, *#nom_station*, specialite, nb_participants)
*Cette relation donne le numéro, la date d'une compétition, la spécialité concernée (ex. alpin, fond), le nom de la station où elle est organisée ainsi que le nombre de participants (skieurs mais aussi délégations, spectateurs…)*

**Classement** (***#no_comp* , *#no_skieur*** , rang)
*Cette relation donne pour chaque skieur participant à une compétition, son rang (ou son classement) dans cette compétition (ex. 1, 2, 3…).*

## Partie I : SQL

1) Créer les schémas des 4 tables de la base de données avec les différentes contraintes en exécutant les requêtes suivantes :

```
CREATE TABLE Station (nom_station varchar(50) Primary key,
alt_station integer,
pays_station varchar(50),
capacite integer);

CREATE TABLE Skieur (no_skieur integer Primary key check (no_skieur > 0),
nom_skieur varchar(50),
prenom varchar(50),
```

```
nom_station varchar(50) references station(nom_station));

CREATE TABLE Competition (no_comp integer primary key check (no_comp > 0),
date_comp DATE, nom_station varchar(50) references station(nom_station),
specialite varchar(50) check (specialite in ('alpin','acrobatique','fond','snowboard')),
nb_participants integer);

CREATE TABLE Classement (no_comp integer references competition(no_comp),
no_skieur integer references skieur(no_skieur),
rang integer DEFAULT 0, primary key(no_comp,no_skieur));
```

2) Insérer les tuples suivants dans la base de données :

```
insert into station values ('Tignes',2000,'France', 250);
insert into station values ('Bormio',3200,'Italie', 150);
insert into station values ('Arvieux en Queyras',1500,'France', 300);
insert into station values ('Ascou Pailheres',2500,'France', 200);

insert into skieur values (1, 'Durand','Olivier','Tignes');
insert into skieur values (2, 'Vogl','Marine','Bormio');
insert into skieur values (3, 'Renaut','Francois','Arvieux en Queyras');
insert into skieur values (4, 'Dupont','Laurent','Ascou Pailheres');
insert into skieur values (5, 'Enault','Francois','Bormio');

insert into competition values (1,'12-01-18', 'Tignes','alpin',120);
insert into competition values (2,'02-28-18', 'Bormio','fond',100);
insert into competition values (3,'02-03-17', 'Tignes','alpin',56);
insert into competition values (4,'01-20-17', 'Tignes','alpin',230);

insert into classement values (1,1,30);
insert into classement values (2,2,12);
insert into classement values (3,1,3);
insert into classement values (4,4,45);
insert into classement values (2,5,2);
insert into classement values (1,3,15);
insert into classement values (3,4,21);
insert into classement values (4,1,15);
insert into classement values (1,2,25);
insert into classement values (3,5,10);
```

Répondre aux questions suivantes en langage SQL en évitant les doublons dans les réponses (**schéma de la relation résultat donné entre parenthèses**).

3) Afficher les noms de colonnes de la table **Competition** qui ne sont pas associés à des contraintes en interrogeant le schéma d'information (information_schema), un ensemble de vues contenant des informations sur les objets définis dans la base de données courante : https://docs.postgresql.fr/14/information-schema.html ***(nom_colonne)***
4) Retrouver la liste des attributs qui ont une valeur par défaut en interrogeant le schéma d'information : https://docs.postgresql.fr/14/information-schema.html ***(nom_colonne)***
5) Afficher les stations dont la valeur d'altitude est supérieure à la valeur moyenne des altitudes de toutes les stations. ***(nom_station)***
6) Afficher les numéros et les dates des compétitions organisées dans une station dont l'altitude est supérieure à 3000 mètres. ***(no_comp, date_comp)***
7) Afficher les stations françaises sur lesquelles il n'y a pas eu de compétition. ***(nom_station)***
8) Afficher les noms des skieurs dont le nom se termine par la lettre 't' ou comportant exactement 4 caractères. L'affichage se fait par ordre alphabétique sur le nom. ***(nom_s)***
9) Quels skieurs ont participé seulement à des compétitions organisées dans leur station d'origine ? ***(nom, prénom)***
10) Quel est le meilleur classement de chaque skieur français ? ***(no_skieur, meilleur_rang)***
11) Quelles stations n'accueillent aucune compétition ? ***(nom_station, pays_station)***
12) Donner le nombre de skieurs de chaque station. ***(nom_station, nb_skieurs)***
13) Supprimer de la table Skieur les skieurs qui ont participé à des compétitions dont le nombre de participants est supérieur à 500.
14) Créer une vue ***VueCompetition*** calculant pour chaque compétition, le nombre de places restantes défini comme la différence entre la capacité de la station et le nombre de participants. On suppose qu'une station n'accueille qu'une compétition à la fois et que la capacité d'accueil est dévolue à cette compétition. ***(no_comp, date_comp, places_libres)***
15) Afficher le contenu actuel de la vue ***VueCompetition***, limité aux attributs no_comp et places_libres. Trier par ordre décroissant du nombre de places libres.

## Partie II : PL/pgSQL

16) Écrire une procédure nommée **ajout_participant** qui permet d'ajouter N participants à une compétition dont le numéro est donné en paramètre. N est un autre paramètre de la procédure. La procédure affiche le nouveau nombre de participants et le nombre de places restantes.

**Contraintes :**

- Utiliser le typage dynamique pour typer les variables et les paramètres de la procédure.
- Traiter correctement au moins l'exception correspondant au dépassement de la capacité de la station correspondante.

17) Tester la procédure **ajout_part** en ajoutant 20 participants à la compétition numéro 1.

18) Écrire une procédure nommée **nbParticipants** qui permet d'afficher les nombres de participants par station et par spécialité.
Exemple d'affichage :

*Station Tignes - Spécialité alpin - nombre de participants : 112*
*Station Tignes - Spécialité fond - nombre de participants : 211*

19) On s'intéresse à la saison complète de ski alpin uniquement. Écrire une fonction nommée **classement_general** prenant en argument l'identifiant d'un skieur et une année et qui renvoie le classement moyen du skieur sur les compétitions de ski alpin de l'année en question. On ne s'intéresse pas au fait que des skieurs aient participé à des nombres différents de compétitions, seule la moyenne des classements compte.

20) Écrire une procédure **podium_general** qui prend en argument deux années et affiche, pour toutes les années, les trois meilleurs skieurs au classement général de ski alpin qui sont présents dans la base de données.
Exemple d'affichage : **podium_general(2017,2019)**

*Coupe 2017 - 1. Durand 2. Dupont 3.Martinet*
*Coupe 2018 - 1. Vogl 2. Hetel 3.Martin*
*Coupe 2019 - 1. Kiki 2. Titi 3.Toto*

21) Tester votre procédure en affichant le podium général des années 2017 et 2018