Saé 2.04: rapport

Table of Contents

[logo] | *logo2.png*

Les participants aux jeux olympiques

Comprendre les données

Exercices(s) 1. Comprendre les données;

Tout d'abord nous avons du analyser les différentes données du fichier csv, par exemple le nombre de lignes contenues dans le csv grâce à la commande :

```
wc -l + nom du fichier
```

nom du fichier	**nombre de lignes**
athlete_events.csv	271 117
noc_regions.csv	231 (mais la commande affiche 0 ligne)

Nous avons traité differentes informations du csv pour qu'ensuite nous puissions anticiper les informations provenant des tables athlete_events et noc_regions comme par exemple faire la disctinction entre les informations et l'entete du fichier :

```
head -n 1 athlete_events.csv
```

```
La commande "head" permet de sélectionner les premières lignes du fichier et "-n 1"
indique le nombre de lignes qu'il faut sélectionner.
```

Pour trouver le séparateur de champs il suffit de regarder le fichier et dans notre cas le séparateur est une virgule, ce séparateur permet de délimiter chaque colonnes du tableau.

Dans notre fichier, une ligne correspond aux informations sur les performances réalisées par un athlète, elles sont représentées sous cette forme :

athlete_events.csv :

nom de la colonne	**Description de la colonne**
id	numéro d'identification de l'athlète (le numéro est unique)
name	nom de l'athlète
sex	sexe de l'athlète homme, femme ou autre (F, M ou null)
age	âge de l'athlète lors de la participation
height	taille de l'athlète lors de la participation
weight	poids de l'athlète lors de la participation
team	nom du pays de l'athlète
noc	numéro d'identification du pays (le numéro est unique pour chaque pays)
games	date et lieu de l'épreuve
year	année où l'epreuve a lieu
season	la saison (Summer soit Winter)
city	nom de la ville où l'épreuve a lieu
sport	nom du sport
event	nom de l'événement
medal	résultat de l'athlète lors de sa participation aux JO (Gold, Silver, Bronze ou null)

```
head -n 1 athlete_events.csv | tr "," "\n" | wc -l"
```

Cette commande sert à compter le nombre de colonnes du fichier athlete_events.csv. Nous

avons commencé par sélectionner la première ligne du fichier avec head -n 1 athlete_events.csv. Puis nous avons remplacé chaque virgule par un retour à la ligne avec la commande tr "," "\n" ce qui nous a permis de simplifier le comptage des colonnes. Enfin, pour compter le nombre de colonnes contenue dans le fichier on a utilisé la commande wc -l qui nous a retourné 15 colonnes.

La colonne qui permet de savoir si les jeux ont lieu en été ou en hiver est la colonne **Season** .

Pour trouver le nombre de lignes qui font référence à **"Jean-Claude Killy"**, nous avons utilisé la commande **grep** qui permet de compter le nombre de fois où une chaîne de caractère est présente dans le fichier. Nous avons donc utilisé la ligne de commande suivante :

```
grep -c "Jean-Claude Killy" athlete_events.csv
```

Ce qui nous a donné 6 lignes.

Maintenant pour connaître le type d'encodage d'un fichier, il faut écrire :

```
file athlete_events.csv -i
```

Enfin nous envisageons de créer une table temporaire nommée import pour importer les données dont nous aurons besion avec cette commande:

```
\copy import from athlete_events.csv with (format csv, delimiter ',', HEADER, NULL
AS 'NA');
```

Cette commande permet de copier les données du fichier “athlete_events.csv” dans la table import. Dans cette commande, on précise le délimiteur pour bien mettre les bonnes données dans les bonnes colonnes et on remplace également les valeurs "NA" par “null”.

Importation des données

Exercices(s) 2. Importation de données

Dans cet exercice nous avons créé une table temporaire qui a la même structure de données que le fichier athlete_events.csv pour ensuite l'importer.

création de la table import :

```
CREATE temporary TABLE import(
    id int,
    name varchar(108),
    sex varchar(1),
    age int,
    height int,
```

```
    weight float,
    team varchar(47),
    noc varchar(3),
    game varchar(11),
    year int,
    season varchar(6),
    city varchar(22),
    sport varchar(25),
    event varchar(85),
    medal varchar(6)
);
\copy import from athlete_events.csv with (format csv, delimiter ',', HEADER, NULL
AS 'NA');
```

Cependant, il y avait des données incorrectes dans le fichier csv que nous avons du supprimer. On a donc effacé toutes les lignes dont la date était inferieure à **1920** et toutes les lignes dont la catégorie de sport était **"artistique"**.

Supression des données incorrectes :

```
DELETE FROM import
WHERE year < 1920 or sport Like ('Art%');
```

Enfin, nous avons du importer le fichier noc_regions.csv dans une table permanente nommée regions pour nous aider dans la suite du projet.

Création de la table regions :

```
CREATE TABLE regions(
    noc varchar(3) PRIMARY KEY,
    region varchar(32),
    notes varchar(27)
);
\copy regions FROM noc_regions.csv with (format csv, delimiter ',', HEADER);;
```

Requêtage sur les tables de départ (import et regions)

Exercices(s) 3. requêtage sur les tables de départ (import et regions)

Dans l'exercice 3, nous avons du réfléchir à des requetes de depart comme connaître le nombre de colonnes dans la table import:

Combien de colonnes dans import :

```
select count(*) as nb_colonnes_import
from INFORMATION_SCHEMA.COLUMNS
```

```
where TABLE_NAME = 'import';
```

Combien de lignes dans import :

```
select count(*) as nb_lignes_import
from import;
```

Combien de codes NOC dans noc :

```
select count(noc) as nb_codes_noc
from regions;
```

Combien d'athlètes différents sont référencés dans ce fichier :

```
select count(distinct id) as nb_athletes
from import;
```

Combien y-a t-il de médailles d'or dans ce fichier :

```
select count(*) as nb_medailles_or
from import where medal = 'Gold';
```

Retrouvez Carl Lewis ; Combien de lignes se réfèrent à Carl Lewis :

```
select count(*) as nb_ligne_carl_lewis
from import
where name like '%Carl% %Lewis%';
```

Ventiler les données

Exercices(s) 4. Ventiler les données

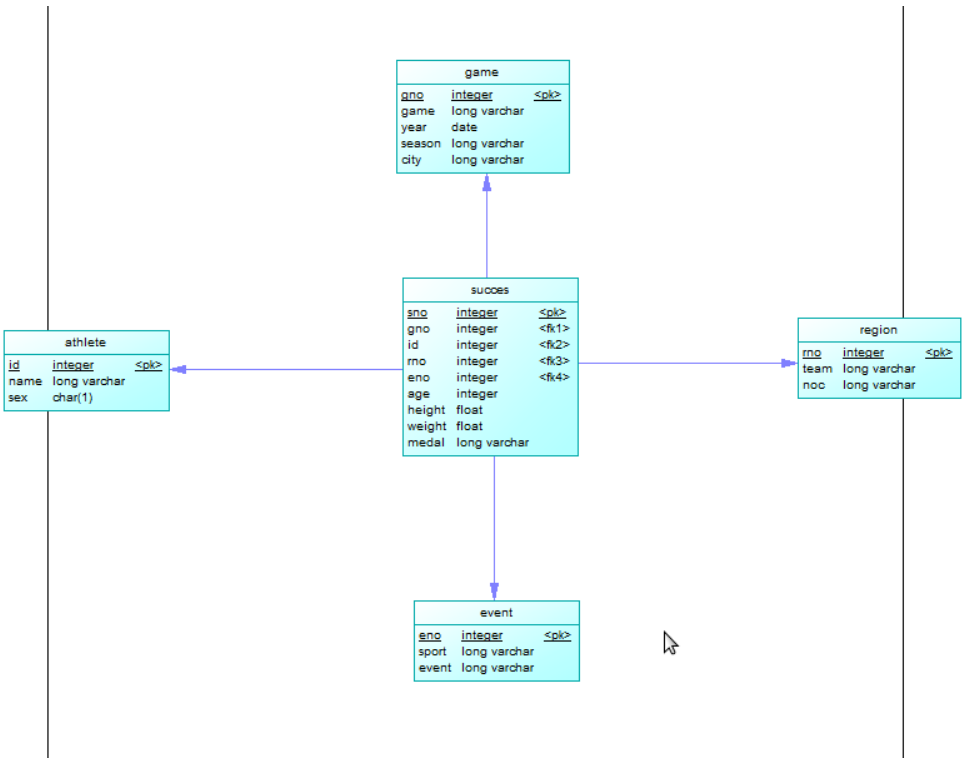

Image du MLD

- athlete(id (pk), name, sex);
- events(eno (pk), sport, event);
- games(gno (pk), year, season, city);
- regions(noc (pk), team, notes);
- succes(sno (pk), gno (fk), id (fk), rno (fk), eno (fk), age, height, weight, medal);

Attention, on a apporté de légères modifications au MLD, notamment dans les tables games et regions, et qu'on a pas pu modifier sur power AMC, pour cela voir la liste des tables qu'on a rappelé ci-dessus.

Pour créer ces tables nous avons réfléchi concrétement au contenu de notre table import.

Lors de la création des tables nous avons eu une incohérence entre les deux tables pour la ville de Singapour, alias Singapore dans les fichiers csv, qui ne possède pas le même noc (SIN pour le fichier noc_regions.csv et SGP pour la table athlete_events.csv) nous avons réglé le problème avec l'ordre sql suivant

Update de la table :

```
update regions set noc = 'SGP' where noc = 'SIN';
```

Partie n°2 :

Pour connaître la taille en octets du fichier **"data-olympique.zip"** il faut faire la commande

suivante :

```
wc -c data-olympique.zip | cut -d ' ' -f 1
```

Ce qui nous donne **5 544 725 octets.**

Pour connaître la taille en octets de la table import, il faut faire la commande suivante :

```
SELECT pg_total_relation_size('import');
```

Ce qui nous donne **47 292 416 octets.**

Pour la table regions, il faut faire la même chose en changeant le nom de la table :

```
SELECT pg_total_relation_size('regions');
```

Ce qui donne **57 344 octets.**

Pour connaître la taille en octets de la somme des tables il faut simplement faire comme avant mais en ajoutant des unions entre les tables, ce qui donne :

```
SELECT pg_total_relation_size('import') +
    pg_total_relation_size('regions') +
    pg_total_relation_size('athletes') +
    pg_total_relation_size('games') +
    pg_total_relation_size('events') +
    pg_total_relation_size('performances') as "Taille total";
```

La taille de la somme des tables est de **75 341 824** octets avec la table import et de **28 049 408** octets sans celle-ci.

Pour la somme des tailles des fichiers exportés, on exporte dans un premier temps nos fichiers, ici “succes, events, athlete, games” pour ce faire :

```
\copy performances TO 'performances.csv' WITH DELIMITER ',' CSV HEADER;
\copy events TO 'events.csv' WITH DELIMITER ',' CSV HEADER;
\copy athletes TO 'athletes.csv' WITH DELIMITER ',' CSV HEADER;
\copy games TO 'games.csv' WITH DELIMITER ',' CSV HEADER;
\copy regions TO 'regions.csv' WITH DELIMITER ',' CSV HEADER;
```

Qui nous donne :

Nom du fichier	Taille (en octets)
athletes.csv	3 633 207

events.csv	28 597
games.csv	1 150
region.csv	3 825
succes.csv	7 047 698

Requêtage

Exercices(s) 5. Requêtage

Pour cet exercice, le but était de faire des requêtes pour tester notre structure de données, mais aussi pour avoir un autre visuel et être plus précis sur les donnéess.

requete 1 :

```
select r.team, count(e.eno) as participation
from regions as r
natural join (succes as s natural join events as e)
group by r.team
order by count(e.eno) desc;
```

requete 2 :

```
select r.team, count(s.medal) as nb_medailles_or
from regions as r
natural join succes as s
where s.medal = 'Gold'
group by r.team
order by count(s.medal) desc;
```

requete 3 :

```
select r.team, count(s.medal) as nb_medailles_totales
from regions as r
natural join succes as s
group by r.team
order by count(s.medal) desc;
```

requete 4 :

```
select id, name, count(medal) as nb_medailles_or
from athlete
natural join succes
where medal = 'Gold'
```

```
group by id, name
having count(medal) >= all(select count(medal) from succes where medal = 'Gold'
group by id, name);
```

requete 5 :

```
select team, count(medal) as nb_medailles_totales
from regions
natural join (succes natural join games)
where city = 'Albertville'
group by team
order by count(medal) desc;
```

requete 6 :

```
select count(distinct s1.id) as nb_sportifs
from regions as r1
natural join (succes as s1 natural join games as g1)
where r1.team != 'France' and exists (select * from regions as r2 natural join
(succes as s2 natural join games as g2) where s1.id = s2.id and g1.year < g2.year
and r2.team = 'France');
```

requete 7 :

```
select count(distinct s1.id) as nb_sportifs
from regions as r1
natural join (succes as s1 natural join games as g1)
where r1.team = 'France' and exists (select * from regions as r2 natural join
(succes as s2 natural join games as g2) where s1.id = s2.id and g1.year < g2.year
and r2.team != 'France');
```

requete 8 :

```
select age, count(medal) as nb_medailles_or
from succes
where medal = 'Gold'
group by age
order by age;
```

requete 9 :

```
select sport, count(medal) as nb_medailles
from events
natural join succes
where age > 50
```

```
group by sport
having count(medal) > 0
order by count(medal) desc;
```

requete 10 :

```
select count(eno) as nb_epreuves, season, year
from games
natural join (succes natural join events)
group by season, year
order by year;
```

requete 11 :

```
select count(medal) as nb_medailles_feminines, year
from games
natural join (succes natural join athlete)
where sex = 'F' and season = 'Summer'
group by year
order by year;
```

Personnalisation du rapport

Exercices(s) 6. Personnalisation du rapport

Pour ce dernier exercice, nous devions créer nos propres requêtes selon un sport et un pays de nôtre choix, alors nous avons pris le "**badminton**" pour le sport et les "**Etats-Unis**" pour le pays.

Pour nos quatre requêtes, nous avons eu les idées suivantes :

Requête 1 : Donner le nombre total de médailles totales obtenues au badminton par les Etats-Unis.

```
select count(medal) as nb_medailles
from succes
where noc in (select noc from regions where team = 'USA') and eno in (select eno
from events where sport = 'Badminton');
```

Requête 2 : Donner la moyenne d'âge de tous les joueurs de badminton aux Etats-Unis.

```
Select avg(age) as moyenne_age
from succes
where noc = 'USA' and eno in (select eno from events where sport = 'Badminton');
```

Requête 3 : Donner les athlètes qui sont les plus grands des Etats-Unis et qui jouent au badminton.

```
select id, name
from athlete
where id in (select id from succes where noc = 'USA' and eno in (select eno from
events where sport = 'Badminton') group by id,height having height >= all(select
height from succes where noc = 'USA' and eno in (select eno from events where
sport = 'Badminton')));
```

Requête 4 : Donner les athlètes qui ont 28 ans, qui font du badminton et qui sont aux Etats-Unis.

```
select id, name
from athlete
where id in (select id from succes where age = 28 and noc = 'USA' and eno in
(select eno from events where sport = 'Badminton'));
```

Conclusion

Pour conclure, on peut dire que grâce à la ventilation, nous avons divisé les deux fichiers csv en différentes tables. Ce qui a permis de réduire la taille en octets des données à manipuler. De plus, grâce aux requêtes ça permet d'avoir une meilleure vue d'ensemble et de mieux ordonner les données.

sources

stackoverflow

modern-sql