

Li  
Olivier

Euros  
BUT1 Informatique

# RAPPORT

SAE S104 - Création d'une base de données: Freedom in the world

## **Sommaire**

### **I - Script manuel de création de la base de données**

### **II - Modélisation et script de création avec *AGL***

- 1 - Réalisation d'une association fonctionnelle
- 2 - Réalisation d'une association maillée
- 3 - Modèle physique de données réalisées
- 4 - Fournir Script SQL donnée par l'AGL
- 5 - Comparaison Script manuel/AGL (figure 2)

### **III - Peuplement des tables**

## I - Script manuel de création de la base de données

En premier lieu, notre première tâche à effectuer est la création d'une base de données par un script, en effet nous devons créer les tables se trouvant dans la figure 2, soit:

- **region(region\_code, name)**
- **status(status)**
- **country(id\_country, name, region\_code, is\_ldc)** où il est précisé que region\_code est une clé étrangère faisant référence au schéma de la relation region
- **freedom(id\_country, civil\_liberties, political\_rights, status)** où status et id\_country sont des clés étrangères qui font respectivement références aux schémas de relation de **status** et de **country**

Afin de réaliser cette première étape, nous allons utiliser la commande **CREATE TABLE** pour créer la table ensuite dans la même requêtes, intégrer ses colonnes avec son **bonnes types de données**:

```
CREATE TABLE region(
    region_code INTEGER PRIMARY KEY,
    name VARCHAR
) ;

CREATE TABLE status(
    status VARCHAR PRIMARY KEY
) ;

CREATE TABLE country(
    id_country SERIAL UNIQUE PRIMARY KEY,
    country VARCHAR,
    region_code INTEGER REFERENCES region,
    is_ldc BOOLEAN NOT NULL
) ;

CREATE TABLE freedom(
    id_country INTEGER REFERENCES country,
```

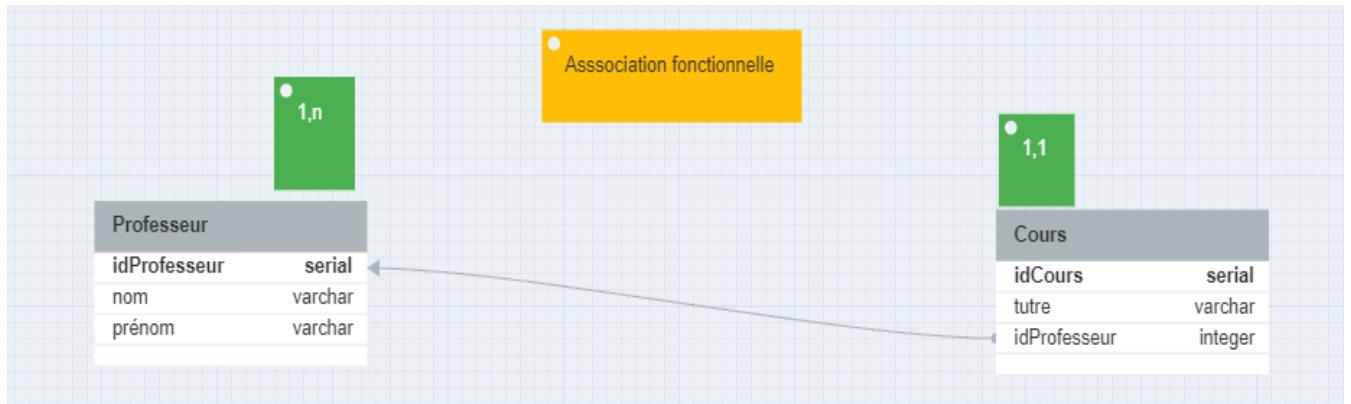
```
year INTEGER,  
civil_liberties INTEGER,  
political_rights INTEGER,  
status VARCHAR REFERENCES status,  
PRIMARY KEY(id_country, year)  
);
```

Nous utilisons:

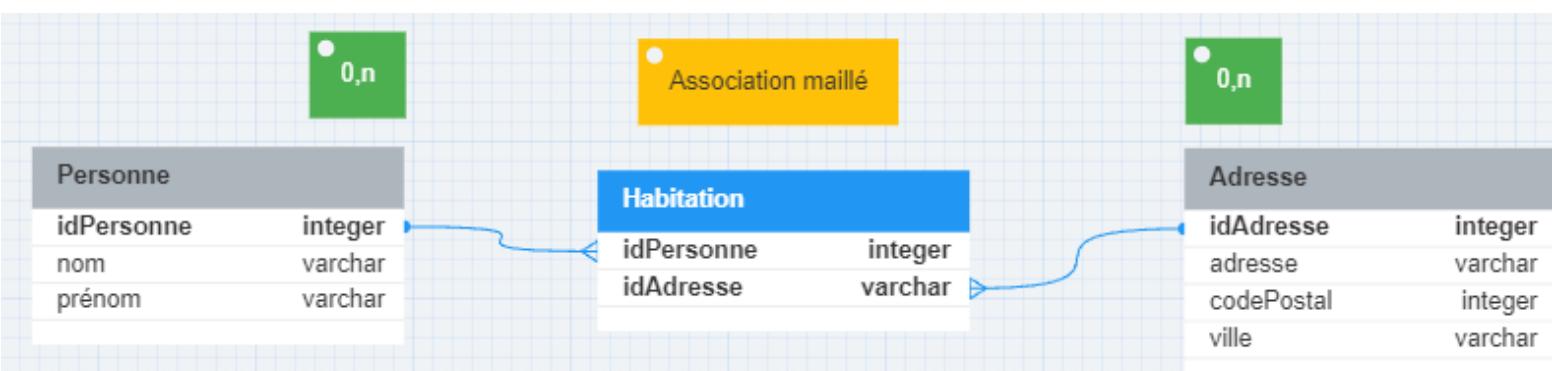
- **INTEGER** pour définir les colonnes et stocker les nombres entiers, il est dans notre cas suivi d'un **REFERENCES** qui indique que la colonne est une clé étrangère.
- **VARCHAR** pour stocker les chaînes de caractères .
- **BOOLEAN** pour stocker les valeurs TRUE et FALSE (is\_Idc).
- **SERIAL** pour créer une colonne auto-incrémentée qui sert de clé primaire dans notre cas (id\_country), assurant un identifiant unique pour chaque enregistrement, elle est complétée par **UNIQUE** pour "renforcer" l'unicité des valeurs et éviter les duplications/doublons.

## II - Modélisation et script de création avec AGL

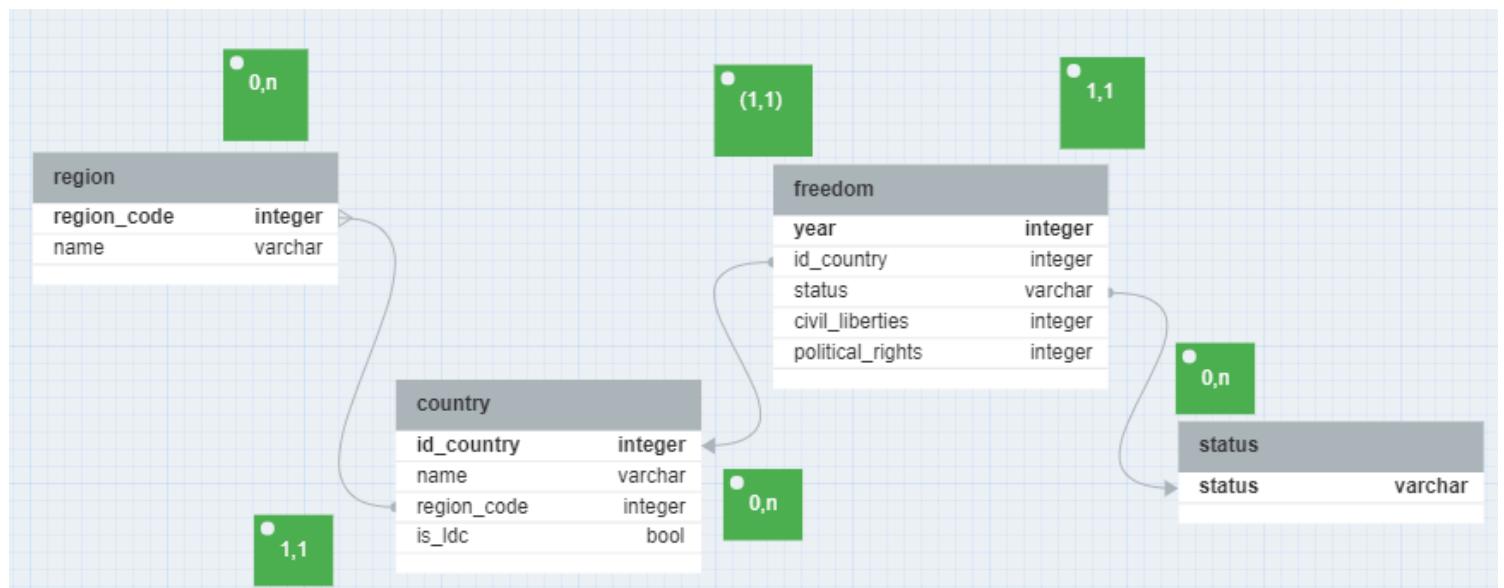
1 - À présent, l'énoncé nous demande dans la partie de la SAE de créer une relation fonctionnelle, soit:



2 - Ensuite d'en réaliser une mais de relation maillée:



3 - On nous demande ensuite de créer le modèle physique sur l'AGL, soit:



4 - Par la suite, on nous demande le script des tables générées automatiquement par l'AGL, soit:

```
region {
    region_code integer pk
    name varchar
}

status {
    status varchar pk unique
}
freedom {
    year integer pk
    id_country integer > country.id_country
    status varchar > status.status
    civil_liberties integer
    political_rights integer
}

country {
    id_country integer pk increments
    name varchar
    region_code integer >* region.region_code
    is_ldc bool
}
```

5 - Et enfin, faire la comparaison entre la création de notre propre script de la figure deux et de celle fourni par l'AGL:

```
region {
    region_code integer pk
    name varchar
} CREATE TABLE region(
    region_code INTEGER PRIMARY KEY,
    name VARCHAR
) ;

status {
    status varchar pk unique
} CREATE TABLE status(
    status VARCHAR PRIMARY KEY
) ;

freedom {
    year integer pk
    id_country integer > country.id_country
    status varchar > status.status
    civil_liberties integer
    political_rights integer
} CREATE TABLE country(
    id_country SERIAL UNIQUE PRIMARY KEY,
    country VARCHAR,
    region_code INTEGER REFERENCES region,
    is_ldc BOOLEAN NOT NULL
) ;

country {
    id_country integer pk increments
    name varchar
    region_code integer >* region.region_code
    is_ldc bool
} CREATE TABLE freedom(
    id_country INTEGER REFERENCES country,
    year INTEGER,
    civil_liberties INTEGER,
    political_rights INTEGER,
    status VARCHAR REFERENCES status,
    PRIMARY KEY(id_country, year)
) ;
```

Comparons:

La table region se distingue par region\_code, utilisée comme clé primaire (pk) pour garantir que chaque région est unique. Elle comprend aussi un name de type **VARCHAR**, permettant une flexibilité dans la longueur des noms des régions.

La table statuts est simple avec une seule colonne, status, qui sert de clé primaire, assurant que chaque statut est distinct et facilement identifiable.

Dans country, id\_country est auto-incrémenté grâce à **SERIAL**, offrant une manière automatique d'assigner un identifiant unique à chaque pays. La colonne region\_code fait référence à region, indiquant une relation de clé étrangère (désigné par >\* pour décrire les relations de plusieurs à plusieurs), is\_Idc lui est un booléen qui requiert une valeur définie, empêchant les erreurs et l'ambiguïté sur le statut de développement du pays.

Et enfin, freedom combine id\_country et year pour former une clé primaire (pk). Cela assure que les données de liberté pour un pays donné sont uniques pour chaque année, permettant un suivi précis des changements au fil du temps.

### **III - Peuplement des tables**

Pour la dernière partie, nous avons comme tâche d'installer un fichier csv qui contient des données sur des populations, en effet nous devons remplir les tables créées manuellement avec ces informations, après avoir fini d'installer il fallait dézipper le fichier csv et le mettre dans un répertoire que nous utilisons (pour mieux se repérer), après celà suite au consigne fourni par M. Audibert, nous devons créer une table temporaire qui va stocker les valeurs du fichier csv puis les renvoyer dans notre propre scripts.

La première chose à faire était de créer la table temporaire, soit:

```
CREATE TABLE change (
    country VARCHAR NOT NULL,
    year INTEGER,
    C_L INTEGER,
    P_R INTEGER,
    status VARCHAR,
    region_code INTEGER,
    region_name VARCHAR,
    is_ldc BOOLEAN
);
```

Ensuite il fallait copier les données du fichier csv dans la table temporaire (change):

```
\COPY change FROM 'freedom.csv' DELIMITER ',' CSV HEADER;
```

après avoir fini nous pouvons vérifier si celà à fonctionner avec:

```
SELECT* FROM CHANGE;
```

À présent, la tâche qu'il nous est conférait est donc de compléter les table de notre propre script :

Pour la table **country**, nous utilisons la commandes suivante:

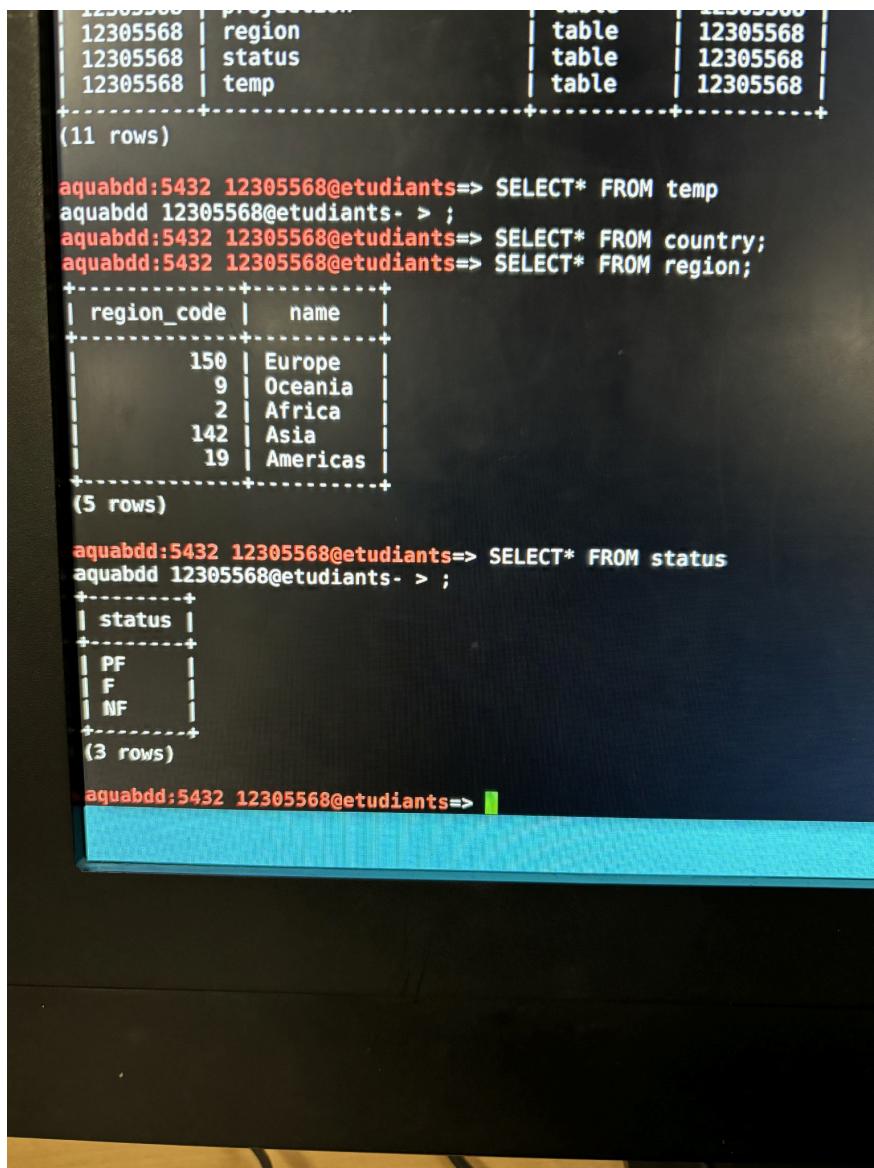
```
INSERT INTO country(country, region_code, is_ldc) SELECT
DISTINCT country, region_code, is_ldc FROM change;
```

```
148 | Israel
149 | Samoa
150 | Mexico
151 | Mauritania
152 | Egypt
153 | Somalia
154 | Namibia
155 | Sudan
156 | Bahamas
157 | Libya
158 | Bosnia and Herzegovina
159 | Turkmenistan
160 | Belize
161 | Argentina
162 | Oman
163 | Netherlands
164 | Georgia
165 | Cuba
166 | Maldives
167 | Central African Republic
168 | Gabon
169 | South Sudan
170 | Slovakia
171 | Australia
172 | Republic of Korea
173 | Russian Federation
174 | Botswana
175 | Liechtenstein
176 | Viet Nam
177 | Venezuela (Bolivarian Republic of)
178 | Costa Rica
179 | Equatorial Guinea
180 | Japan
181 | San Marino
182 | Marshall Islands
183 | Hungary
184 | Djibouti
185 | Iraq
186 | Seychelles
187 | Niger
188 | Singapore
189 | Antigua and Barbuda
190 | Jamaica
191 | Malawi
192 | Lao People's Democratic Republic
193 | Ghana
+
(193 rows)
(END)
```

---

Pour **region** et **status** respectivement:

```
INSERT INTO region SELECT DISTINCT region_code,  
region_name FROM change;
```



```
+-----+-----+-----+-----+  
| 12305568 | region      | table   | 12305568 |  
| 12305568 | status       | table   | 12305568 |  
| 12305568 | temp        | table   | 12305568 |  
+-----+-----+-----+-----+  
(11 rows)  
  
aquabdd:5432 12305568@etudiants=> SELECT* FROM temp  
aquabdd 12305568@etudiants-> ;  
aquabdd:5432 12305568@etudiants=> SELECT* FROM country;  
aquabdd:5432 12305568@etudiants=> SELECT* FROM region;  
+-----+-----+  
| region_code | name    |  
+-----+-----+  
| 150 | Europe |  
| 9 | Oceania |  
| 2 | Africa |  
| 142 | Asia |  
| 19 | Americas |  
+-----+-----+  
(5 rows)  
  
aquabdd:5432 12305568@etudiants=> SELECT* FROM status  
aquabdd 12305568@etudiants-> ;  
+-----+  
| status |  
+-----+  
| PF |  
| F |  
| NF |  
+-----+  
(3 rows)  
  
aquabdd:5432 12305568@etudiants=> █
```

pour **status**:

---

```
INSERT INTO status SELECT DISTINCT status FROM change;
```

---

et enfin pour la table **freedom**:

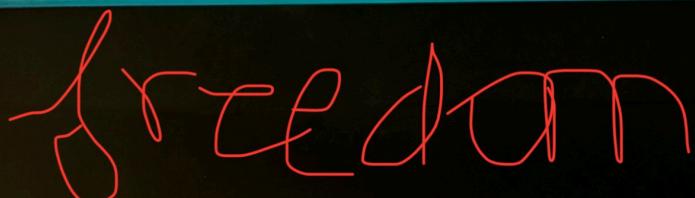
```
INSERT INTO freedom SELECT year, id_country, C_L, P_R,  
status FROM change JOIN country as x USING (country)
```

affichant:

2015	14	3	1	F
2000	14	3	3	PF
2017	27	5	2	F
2012	188	4	5	PF
2009	109	5	4	PF
2010	191	4	6	NF
2010	103	5	3	PF
2019	160	2	5	PF
2010	122	2	2	F
2004	135	5	1	F
2013	137	4	5	PF
2000	96	6	4	PF
2018	98	5	6	NF
2007	129	2	6	PF
2006	191	3	2	F
2003	49	1	4	PF
2011	28	6	1	F
1998	12	2	6	NF
2000	48	2	2	F
1997	119	4	2	F
1997	93	3	4	PF
2018	89	3	4	PF
2008	48	1	3	PF
2009	86	6	1	F
1995	66	6	6	NF
2005	111	1	7	NF
2002	129	3	1	F
1997	148	3	3	PF
2007	71	4	1	F
2010	72	3	5	PF
1997	97	5	3	PF
2002	154	3	7	NF
2011	45	4	2	F
2008	21	5	5	PF
1997	124	1	5	PF
2007	140	1	1	F
2007	125	1	1	F
2007	17	3	1	F
1995	83	4	5	PF
2003	149	2	4	PF
1998	130	1	2	F
			1	F

(4979 rows)

(END)



Nous avons donc fini de transporter les données de la table temporaire au script à présent le script réalisée manuellement contient toutes les informations du fichier csv, le peuplement de table à donc été effectué notre fin d'SAÉ de création d'une base de données s'achève ici.