SAE Base de Données (S104)

**SOMMAIRE :**

-2.1. Script de création de tables

-2.2. Modélisation, comparaison et script de création avec AGL

-2.3. Peuplement des tables

***2.1***

-- Création de la table "region"

**CREATE TABLE region (**

**region\_code INTEGER PRIMARY KEY,**

**name TEXT**

**);**

-- Création de la table "status"

**CREATE TABLE status (**

**status TEXT PRIMARY KEY**

**);**

-- Création de la table "country"

**CREATE TABLE country (**

**id\_country SERIAL PRIMARY KEY,**

**name TEXT,**

**region\_code INTEGER,**

**is\_ldc INT,**

**FOREIGN KEY (region\_code) REFERENCES region(region\_code)**

**);**

-- Création de la table "freedom"

**CREATE TABLE freedom (**

**id\_country SERIAL,**

**year INTEGER,**

**civil\_liberties INTEGER,**

**political\_rights INTEGER,**

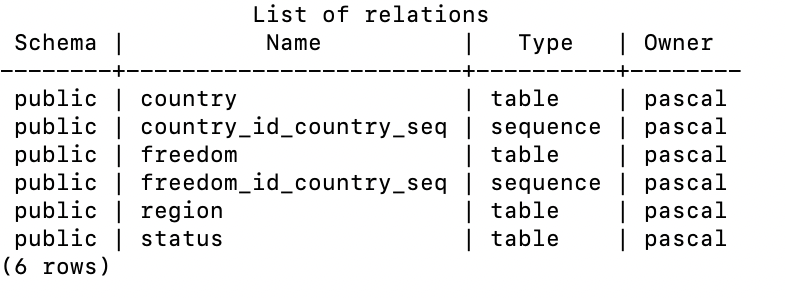
**status TEXT,**

**PRIMARY KEY (id\_country, year),**

**FOREIGN KEY (id\_country) REFERENCES country(id\_country),**

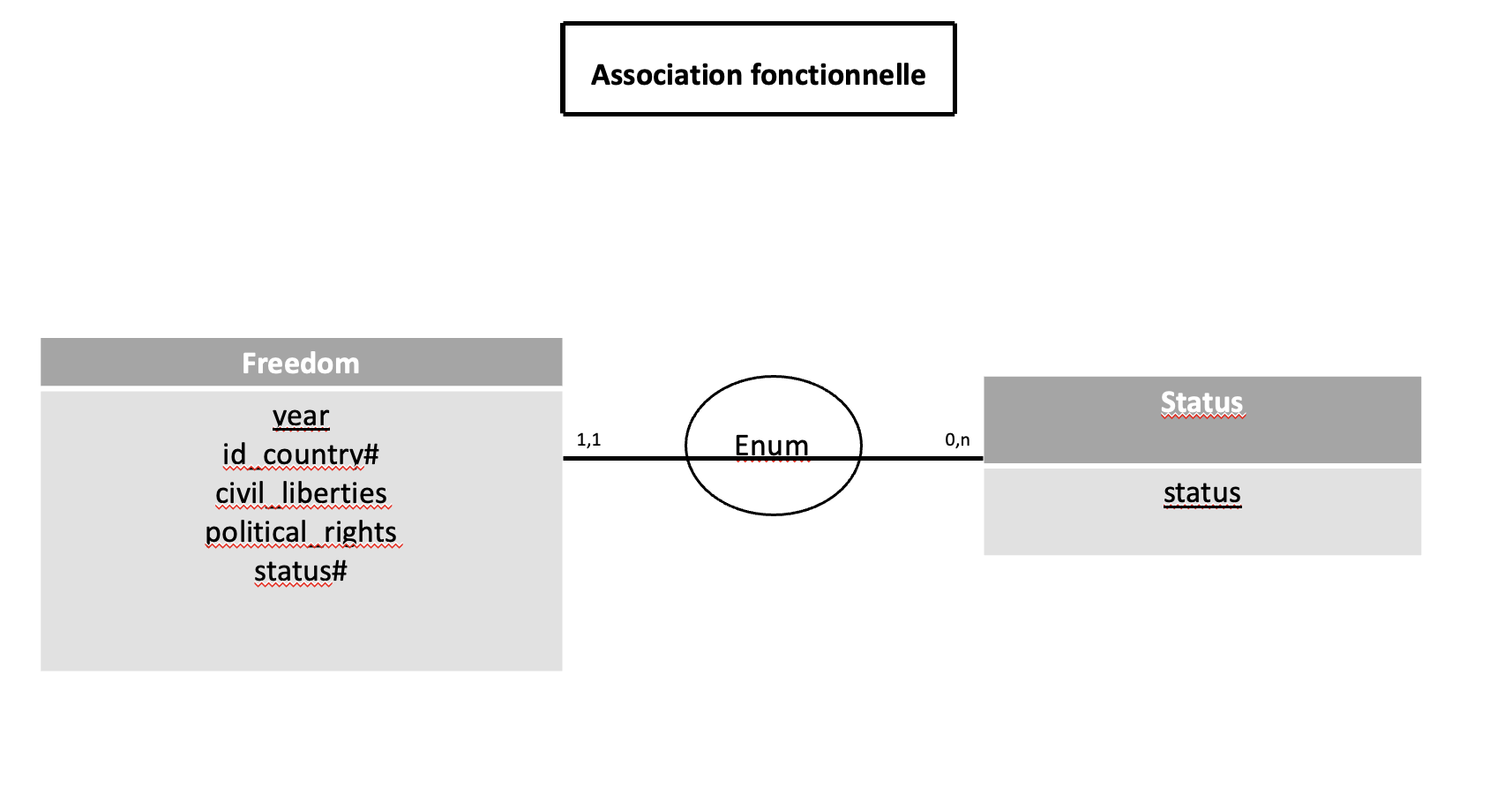
**FOREIGN KEY (status) REFERENCES status(status)**

**);**

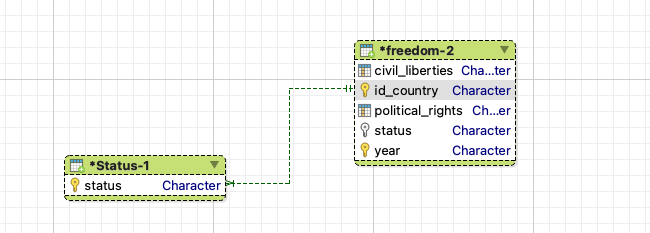


***2.2.1***

*Voici une association fonctionnelle sous la forme d’un modèle conceptuel de données :*



*Voici une association fonctionnelle sous la forme d’un modèle physique de données :*

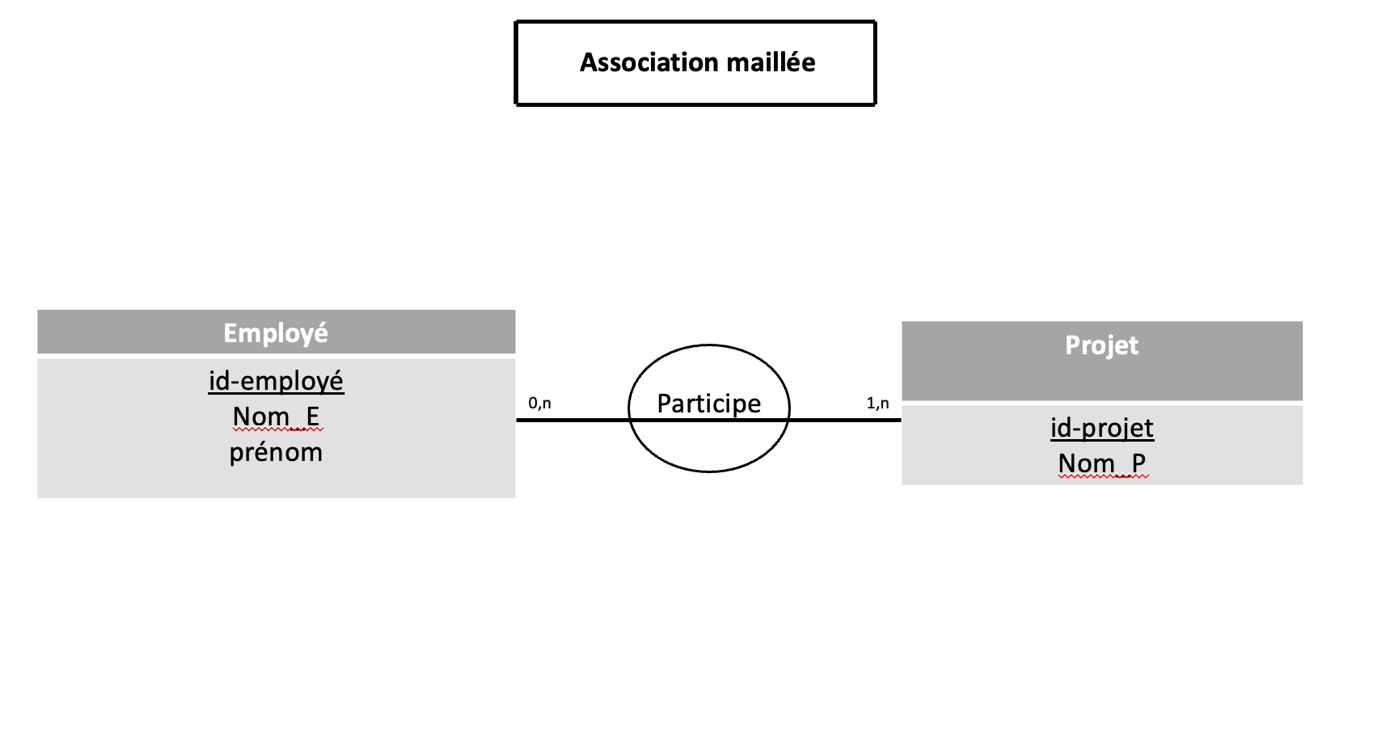


Enum

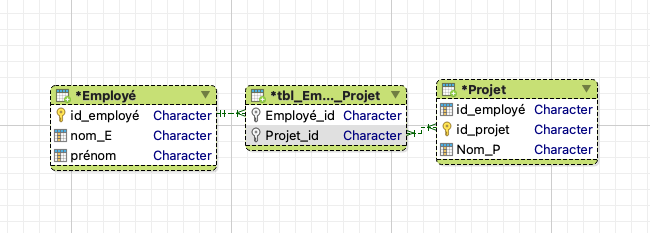
Un modèle conceptuel de données (MCD) est un modèle de données qui représente les données de manière abstraite. Ainsi, on voit qu’en passant à un modèle physique de données (MPD), les entités sont transformées en tables afin d’en augmenter l’utilisabilité et la performance. On passe alors d’une représentation abstraite à une représentation concrète. En effet, cette transformation permet de prendre en compte les contraintes technologiques des bases de données relationnelles, comme par exemple la nécessité d’utiliser des clés primaires et étrangères afin de garantir l’intégrité des données. De ce fait, on voit des différences entre le MCP et le MPD. Tout d’abord, on voit que les cardinalités, présentes sur le MCD, sont remplacées par des symboles en bout de liaisons pour le MPD. Ensuite, sur le MPD, les clés primaires sont ici explicitement notées avec un symbole de clé doré et les clés étrangères le sont par un symbole de clé argentée. Cela permet donc de mieux se repérer. Enfin, pour le MPD, les types des champs sont clairement indiqués, ce qui permet donc une meilleure compréhension.

***2.2.2***

*Voici une association maillée sous la forme d’un modèle conceptuel de données :*



*Voici une association maillée sous la forme d’un modèle physique de données :*

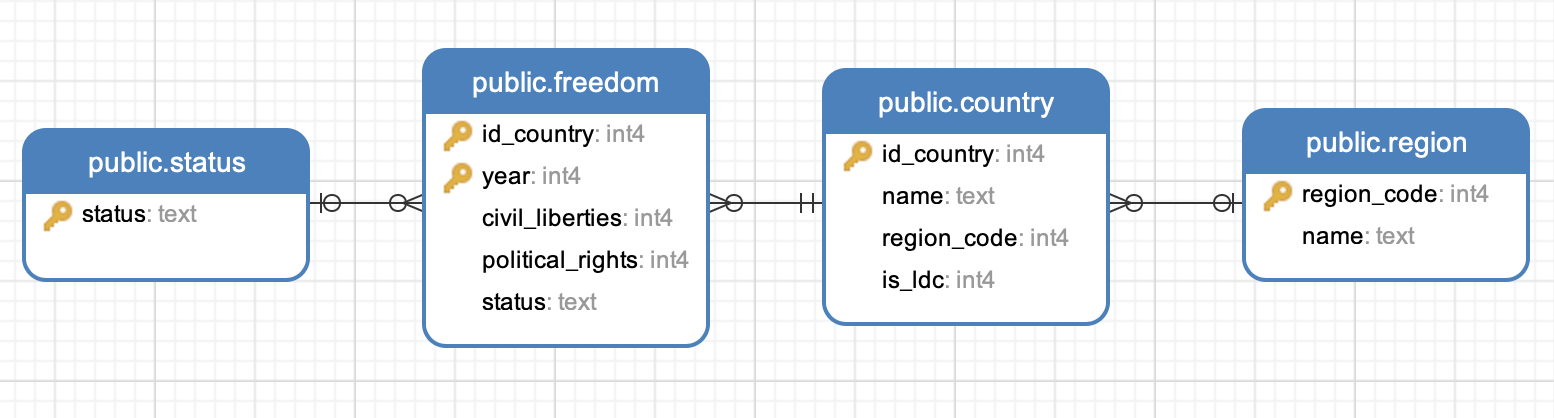


L’association maillée diffère également suivant le modèle choisi. Dans un premier temps, on voit bien que le type association qui relie les deux entités dans le MCD est remplacé par une table intermédiaire dans le MPD. Cela permet de garantir l’intégrité des données, en évitant notamment la présence de doublons, ou bien la perte de données. Ensuite, les types des champs sont indiqués de manière explicite dans le MPD, ce qui permet de savoir quel type de données doit être stocké dans chaque champ. De plus, les clés primaires et étrangères sont notées respectivement par un symbole de clé dorée et de clé argentée, ce qui permet de simplifier le modèle et de e rendre plus facile à comprendre. Enfin, les cardinalités, qui étaient explicitées dans le MCD, sont transformées en symboles placés en bout de liaison pour le MPD.

***2.2.3***

L’AGL que j’ai choisi est Navicat.

En recréant sur cet AGL le modèle physique de données correspondant au modèle conceptuel de données représenté sur la figure 2, on obtient ce modèle :

****

***2.2.4***

CREATE TABLE "country" (

"id\_country" int4 NOT NULL DEFAULT nextval('country\_id\_country\_seq'::regclass),

"name" text COLLATE "pg\_catalog"."default",

"region\_code" int4,

"is\_ldc" int4,

CONSTRAINT "country\_pkey" PRIMARY KEY ("id\_country")

);

ALTER TABLE "country" OWNER TO "lucas";

CREATE TABLE "freedom" (

"id\_country" int4 NOT NULL DEFAULT nextval('freedom\_id\_country\_seq'::regclass),

"year" int4 NOT NULL,

"civil\_liberties" int4,

"political\_rights" int4,

"status" text COLLATE "pg\_catalog"."default",

CONSTRAINT "freedom\_pkey" PRIMARY KEY ("id\_country", "year")

);

ALTER TABLE "freedom" OWNER TO "lucas";

CREATE TABLE "region" (

"region\_code" int4 NOT NULL,

"name" text COLLATE "pg\_catalog"."default",

CONSTRAINT "region\_pkey" PRIMARY KEY ("region\_code")

);

ALTER TABLE "region" OWNER TO "lucas";

CREATE TABLE "status" (

"status" text COLLATE "pg\_catalog"."default" NOT NULL,

CONSTRAINT "status\_pkey" PRIMARY KEY ("status")

);

ALTER TABLE "status" OWNER TO "lucas";

ALTER TABLE "country" ADD CONSTRAINT "country\_region\_code\_fkey" FOREIGN KEY ("region\_code") REFERENCES "region" ("region\_code") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "freedom" ADD CONSTRAINT "freedom\_id\_country\_fkey" FOREIGN KEY ("id\_country") REFERENCES "country" ("id\_country") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "freedom" ADD CONSTRAINT "freedom\_status\_fkey" FOREIGN KEY ("status") REFERENCES "status" ("status") ON DELETE NO ACTION ON UPDATE NO ACTION;

***2.2.5***

Plusieurs différences sont à noter entre mon script initial et celui généré automatiquement par l’AGL. Tout d’abord, en ce qui concerne l’auto incrémentation de la clé primaire de la table ‘*country’*, dans mon script initial j’utilise le mot-clé ***SERIAL***, alors que mon AGL a préféré la syntaxe ***DEFAULT nextval(‘sequence\_name’ ::regclass)***. Cela souligne simplement une variation d’approche pour atteindre le même objectif d’auto incrémentation.

Ensuite, parlons des contraintes de clés primaires. Dans mon script, je déclare la contrainte sans utiliser la syntaxe ***CONSTRAINT***, alors que l’AGL inclut explicitement cette syntaxe avec un nom par défaut. Cela reflète les conventions de déclaration des contraintes de clés primaires spécifiques à mon AGL.

Pour ce qui est des clés étrangères, dans mon script, les clés étrangères sont intégrées dans la définition de la colonne, alors que mon AGL ajoute les contraintes associées après la création de la table. De plus, il y spécifie des options détaillées, telles que ***ON DELETE NO ACTION ON UPGRADE NO ACTION*,** qui permettent d’assurer la sauvegarde des données en cas d’effacement ou de mise à jour des tables.

Comme autre différence, on peut noter la spécification du propriétaire de la table. Mon script n’inclut aucune information sur le propriétaire. Cependant, mon AGL utilise la clause ***ALTER TABLE*** afin de définir le propriétaire après la création de la table.

Ensuite, en ce qui concerne la collation des colonnes de type « TEXT », dans mon script initial, cette information n’est pas incluse, tandis que Navicat utilise la syntaxe COLLATE afin de spécifier cela.

***2.3***

#On commence par créer une table temporaire dont les champs correspondent à tous les champs du fichier csv pour y mettre toutes les données contenues ce fichier csv :

**CREATE TEMPORARY TABLE temp\_data (**

**country TEXT,**

**year INTEGER,**

**CL INTEGER,**

**PR INTEGER,**

**Status TEXT,**

**Region\_Code INTEGER,**

**Region\_Name TEXT,**

**is\_ldc INTEGER**

**);**

#On met les données du fichier csv dans cette table temporaire avec la commande *COPY*:

**COPY temp\_data(country, year, CL, PR, Status, Region\_Code, Region\_Name, is\_ldc)**

**FROM '/Users/lucas/Desktop/Documents\_lucas/BDD/freedom.csv' DELIMITER ',' CSV HEADER;**

#Maintenant, on commence à peupler les tables en prenant les données de la table temporaire pour les faire correspondre avec les champs des différentes tables.

# Les données distinctes des colonnes Region\_Code et Region\_Name de la table temporaire sont insérées dans la table region :

**INSERT INTO region(region\_code, name)**

**SELECT DISTINCT Region\_Code, Region\_Name FROM temp\_data;**

# Les données distinctes de la colonne Status de la table temporaire sont insérées dans la table status :

**INSERT INTO status(status)**

**SELECT DISTINCT Status FROM temp\_data;**

# Les données distinctes des colonnes country, Region\_Code, et la valeur maximale de is\_ldc sont insérées dans la table country. Le regroupement se fait sur les colonnes country et Region\_Code :

**INSERT INTO country(name, region\_code, is\_ldc)**

**SELECT DISTINCT country, Region\_Code, MAX(is\_ldc) FROM temp\_data GROUP BY country, Region\_Code;**

#Ici, on fait une jointure afin de faire correspondre l’id\_country de la table country avec celui de la table freedom :

**INSERT INTO freedom (id\_country, year, civil\_liberties, political\_rights, status)**

**SELECT c.id\_country, t.year, t.CL, t.PR, t.Status**

**FROM temp\_data t**

**JOIN country c ON t.country = c.name AND t.Region\_Code = c.region\_code;**

#Enfin, on efface la table temporaire une fois l’insertion des données dans les autres tables terminée.

**DROP TABLE temp\_data;**

Voici donc le script sans les descriptions et commentaires associés :

**COPY temp\_data(country, year, CL, PR, Status, Region\_Code, Region\_Name, is\_ldc)**

**FROM '/Users/lucas/Desktop/Documents\_lucas/BDD/freedom.csv' DELIMITER ',' CSV HEADER;**

**INSERT INTO region(region\_code, name)**

**SELECT DISTINCT Region\_Code, Region\_Name FROM temp\_data;**

**INSERT INTO status(status)**

**SELECT DISTINCT Status FROM temp\_data;**

**INSERT INTO country(name, region\_code, is\_ldc)**

**SELECT DISTINCT country, Region\_Code, MAX(is\_ldc) FROM temp\_data GROUP BY country, Region\_Code;**

**INSERT INTO freedom(id\_country, year, civil\_liberties, political\_rights, status)**

**SELECT c.id\_country, t.year, t.CL, t.PR, t.Status**

**FROM temp\_data t**

**JOIN country c ON t.country = c.name AND t.Region\_Code = c.region\_code;**

**DROP TABLE temp\_data;**

On obtient les résultats suivants :

