

I U T D E V I L L E T A N E U S E

S A E S 1 0 4

# R A P P O R T

EL MOUTTAQUI Weame

BUT1 Groupe : Némée

# Table des matières

|  |          |
|--|----------|
| <b>Table des matières.....</b>   | <b>2</b> |
| <b>I- Script manuel de création de la base de données .....</b>        | <b>3</b> |
| 2.1 Script manuel .....  | 3        |
| <b>II - Modélisation et script de création avec AGL .....</b>          | <b>4</b> |
| 2.2 .....  | 4        |
| 1.Illustrations comparative cours/ AGL association fonctionnelle ..... | 4        |
| 2. Illustrations comparatives cours/ AGL association maillé .....      | 5        |
| 3. Modèle physique de données avec AGL .....                           | 6        |
| 4. Script SQL de creation des tables générés automatiquement .....     | 6        |
| 5 Discussion sur les différences .....                                 | 8        |
| <b>III - Peuplement des tables .....</b>                               | <b>9</b> |
| 1. Script de peuplement de la base de données .....                    | 9        |
| 2. Description commentée des différentes étapes du script .....        | 10       |

## **I - 2.1 Script manuel de création de la base de données**

**Script réalisé à partir du schéma relationnel :**

```
DROP TABLE IF EXISTS climate_disaster ;
DROP TABLE IF EXISTS country ;
DROP TABLE IF EXISTS sub_region;
DROP TABLE IF EXISTS region;
DROP TABLE IF EXISTS disaster ;
```

```
CREATE TABLE region (
    region_code INTEGER PRIMARY KEY,
    name_region VARCHAR NOT NULL ) ;
```

```
CREATE TABLE sub_region (
    sub_region_code INTEGER PRIMARY KEY,
    name_sub_region VARCHAR NOT NULL,
    region_code INTEGER REFERENCES region (region_code) ON DELETE CASCADE) ;
```

```
CREATE TABLE country (
    country_code INTEGER PRIMARY KEY,
    name_country VARCHAR NOT NULL,
    ISO2 CHAR(2) ,
    ISO3 CHAR(3) NOT NULL,
    region_code INTEGER REFERENCES region (region_code) ON DELETE CASCADE ) ;
```

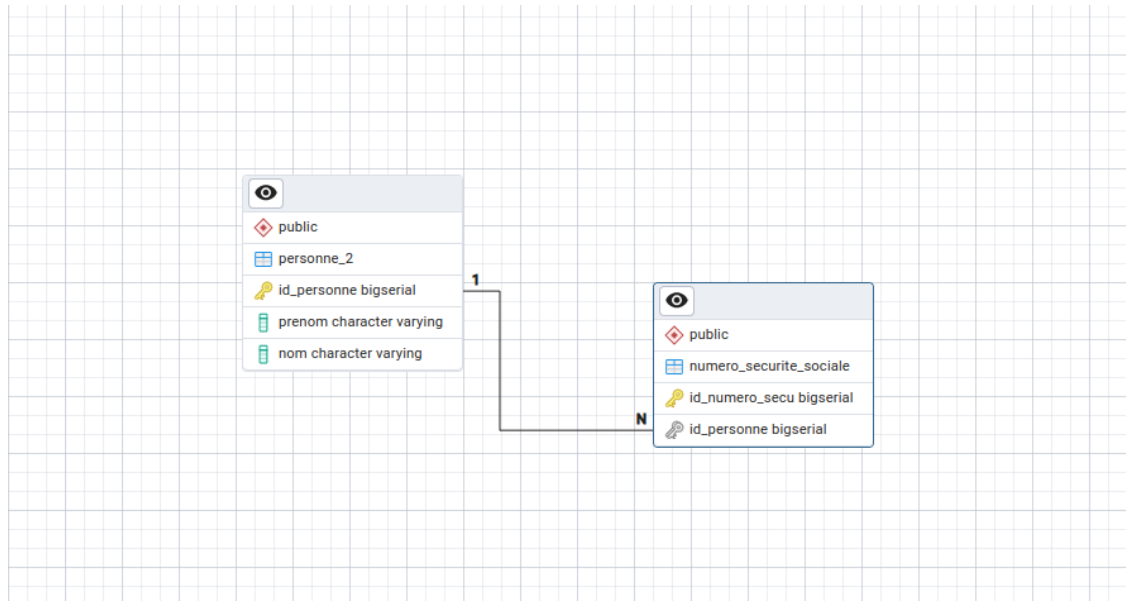
```
CREATE TABLE disaster (
    disaster_code INTEGER PRIMARY KEY,
    disaster VARCHAR NOT NULL ) ;
```

```
CREATE TABLE climate_disaster (
    country_code INTEGER REFERENCES country ( country_code),
    disaster_code INTEGER REFERENCES disaster ( disaster_code) ,
    year INTEGER CHECK ( year > 0 OR year IS NULL ),
    number INTEGER NOT NULL,
    PRIMARY KEY (country_code, disaster_code, year)) ;
```

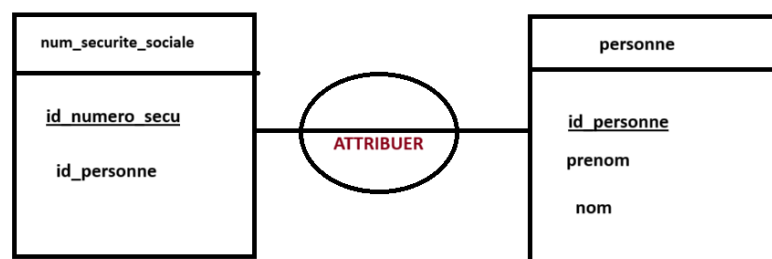
## II - 2.2 Modélisation et script de création avec AGL

Dans le cadre de la SAE, j'ai choisit l'Atelier de Génie Logiciel pgAdmin4 version 8.14.

### 1. Associations fonctionnelles :

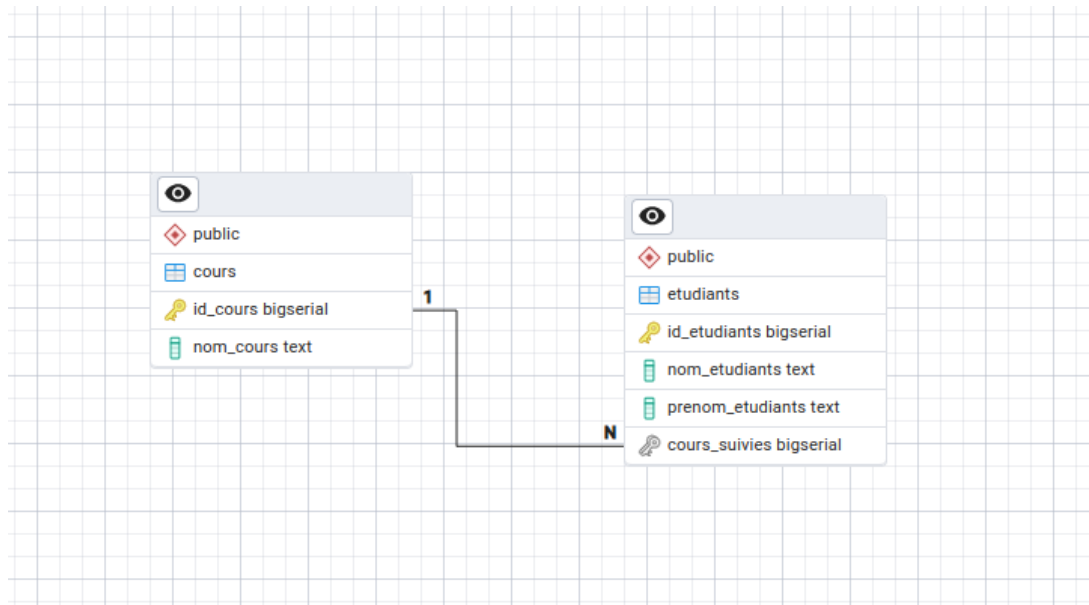


### Modèle entités associations, formalisme AGL pgAdmin4

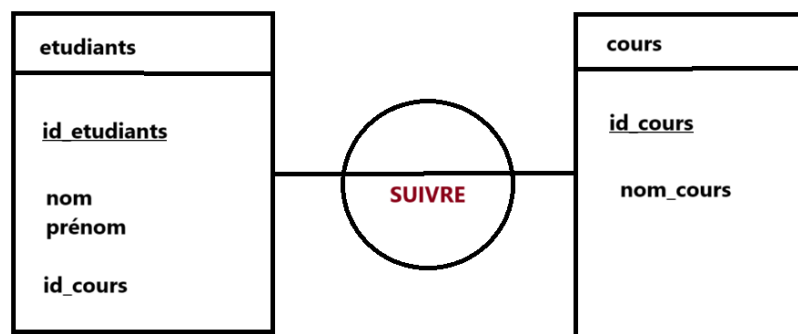


### Modèle entités associations, formalisme vu en cours

## 2. Associations maillées:




Modèle entités associations, formalisme AGL pgAdmin4



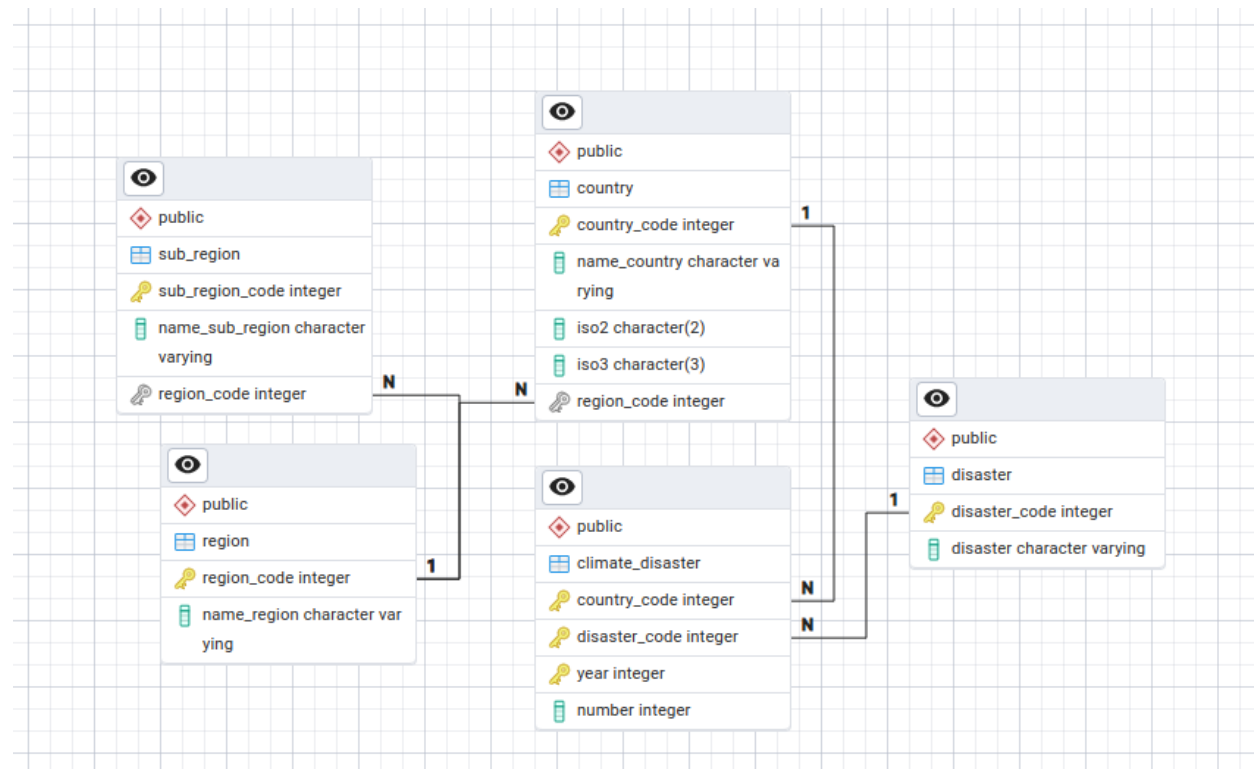
Modèle entités associations, formalisme vu en cours

### Explications :

Avec pgAdmin4 les modèles ne sont pas très différents de ceux vu en cours. Pour le lire, il faut lire le nom de la table à côté de ce sigle . Afin d'identifier la clé primaire, il suffit de chercher le sigle clé doré parmi les cases. Il faut la différencier la clé étrangère qui a pour sigle une clé grise. Pour le reste des attributs ils ont simplement un sigle vert à côté. Concernant la cardinalité elle diffère de celle vu en cours ; Selon ce qui est affiché 1 ou N il est question du

maximum d'occurrences qu'une entité puisse être reliées à une autre entité. Pour finir le type de l'attribut est écrit à la suite de l'attribut ce qui donne plus d'informations.

### 3. Modèle physique de données de la figure 4 réalisé avec l'AGL:



### 4. Script SQL de création des tables généré automatiquement par l'AGL:

```

BEGIN; /* Début du programme généré automatiquement par pgAdmin4 */
DROP TABLE IF EXISTS climate_disaster, country, disaster, region, sub_region CASCADE;

CREATE TABLE IF NOT EXISTS public.climate_disaster (
  country_code integer NOT NULL,
  disaster_code integer NOT NULL,
  year integer NOT NULL,
  "number" integer NOT NULL,
  CONSTRAINT climate_disaster_pkey PRIMARY KEY (country_code, disaster_code, year));

CREATE TABLE IF NOT EXISTS public.country (
  country_code integer NOT NULL,
  name_country character varying COLLATE pg_catalog."default" NOT NULL,

```

```
iso2 character(2) COLLATE pg_catalog."default" NOT NULL,  
iso3 character(3) COLLATE pg_catalog."default" NOT NULL,  
region_code integer,  
CONSTRAINT country_pkey PRIMARY KEY (country_code));
```

```
CREATE TABLE IF NOT EXISTS public.disaster (  
    disaster_code integer NOT NULL,  
    disaster character varying COLLATE pg_catalog."default" NOT NULL,  
    CONSTRAINT disaster_pkey PRIMARY KEY (disaster_code));
```

```
CREATE TABLE IF NOT EXISTS public.region (  
    region_code integer NOT NULL,  
    name_region character varying COLLATE pg_catalog."default" NOT NULL,  
    CONSTRAINT region_pkey PRIMARY KEY (region_code));
```

```
CREATE TABLE IF NOT EXISTS public.sub_region (  
    sub_region_code integer NOT NULL,  
    name_sub_region character varying COLLATE pg_catalog."default" NOT NULL,  
    region_code integer,  
    CONSTRAINT sub_region_pkey PRIMARY KEY (sub_region_code));
```

```
ALTER TABLE IF EXISTS public.climate_disaster  
    ADD CONSTRAINT climate_disaster_country_code_fkey FOREIGN KEY (country_code)  
    REFERENCES public.country (country_code) MATCH SIMPLE  
    ON UPDATE NO ACTION  
    ON DELETE NO ACTION;
```

```
ALTER TABLE IF EXISTS public.climate_disaster  
    ADD CONSTRAINT climate_disaster_disaster_code_fkey FOREIGN KEY (disaster_code)  
    REFERENCES public.disaster (disaster_code) MATCH SIMPLE  
    ON UPDATE NO ACTION  
    ON DELETE NO ACTION;
```

```
ALTER TABLE IF EXISTS public.country  
    ADD CONSTRAINT country_region_code_fkey FOREIGN KEY (region_code)  
    REFERENCES public.region (region_code) MATCH SIMPLE  
    ON UPDATE NO ACTION  
    ON DELETE CASCADE;
```

```
ALTER TABLE IF EXISTS public.sub_region  
    ADD CONSTRAINT sub_region_region_code_fkey FOREIGN KEY (region_code)  
    REFERENCES public.region (region_code) MATCH SIMPLE  
    ON UPDATE NO ACTION
```

ON DELETE CASCADE;

END;            /\* Fin du programme généré automatiquement par pgAdmin4 \*/

## 5. Discussion sur les différences entre les scripts produits manuellement et automatiquement

La première chose que l'on remarque est la longueur des scripts qui diffère. D'un côté le script fait manuellement est court concis avec 30 lignes et de l'autre côté celui produit automatiquement est long avec 50 lignes soit un peu moins du double de l'autre. Celui généré par pgAdmin4 ne commence pas par 'DROP TABLE IF EXISTS' comme je l'ai fait manuellement, mais il remplace plutôt par 'ALTER TABLE IF EXISTS', ou le cas échéant 'CREATE TABLE IF NOT EXISTS'. Le script généré automatiquement commence par un 'BEGIN' et termine par un 'END' ce qui signifie que le script entier est considéré comme une « seule » entité.

Le script fait par l'AGL gère bien mieux les potentiels erreurs que pourrait rencontrer le script avec 'ADD CONSTRAINT' ou 'ON UPDATE NO ACTION'... Tandis que le mien est très simple avec une complexité très faible. L'AGL fait en sorte d'éviter toute sorte d'erreurs pouvant perturber la création des tables.

Finalement, les deux scripts nous permettent d'obtenir le même résultat, mais de manière différente. Le script réalisé à la main convient pour des scénarios simples où il y a un risque peu élevé d'erreurs alors que le script généré par pgAdmin4 est plus robuste et conviendrait à un scénarios plus complexe.

## III - 2.3 Peuplement des tables

### **1. Script de peuplement de la base de données**

✧ Les commentaires du code sont en *gris*.

DROP TABLE IF EXISTS table\_intermediaire ;

```
CREATE TEMPORARY TABLE table_intermediaire (  
    country VARCHAR NOT NULL,  
    iso2 VARCHAR(2),  
    iso3 VARCHAR (3),  
    region_code INTEGER NOT NULL,  
    region VARCHAR,  
    sub_region_code INTEGER NOT NULL,
```



```

        sub_region VARCHAR,
        disaster VARCHAR,
        year INTEGER CHECK (year > 0 or year IS NOT NULL),
        number INTEGER);
\COPY table_intermediaire FROM Climate_related_disasters_frequency.csv WITH CSV
HEADER ENCODING 'UTF8' ;

SELECT * FROM table_intermediaire;          /* Vérification de la table intermédiaire*/

/* DEUXIEME PARTIE INSERTION DANS NOTRE BASE DE DONNÉES */

INSERT INTO region (region_code, name_region)
SELECT DISTINCT region_code, region
FROM table_intermediaire ;          /* on insert les données de la table temporaire dans
la table region */

INSERT INTO sub_region (sub_region_code, name_sub_region, region_code)
SELECT DISTINCT sub_region_code, sub_region, region_code          /* le distinct sert à
éviter les doublons et donc la redondance*/
FROM table_intermediaire ;

INSERT INTO country (country_code, name_country, ISO2, ISO3, region_code)
SELECT ROW_NUMBER() OVER (ORDER BY country) AS country_code, country, iso2, iso3,
region_code          /* cette ligne et (ROW NUMBER()) permet de donner
le code du pays par ordre croissant selon l'ordre alphabétique en renommant la colonne par
country_code*/
FROM (SELECT DISTINCT country, iso2, iso3, region_code FROM table_intermediaire) AS
distinct_countries ;

INSERT INTO disaster (disaster_code, disaster)
SELECT ROW_NUMBER() OVER (ORDER BY disaster) AS disaster_code, disaster
/* cette ligne permet de donner le code de la catastrophe naturelle par ordre
croissant selon l'ordre alphabétique en renommant la colonne par disaster_code*/
FROM (SELECT DISTINCT disaster FROM table_intermediaire) AS distinct_disasters ;

INSERT INTO climate_disaster (country_code, disaster_code, year, number)
SELECT c.country_code, d.disaster_code, ti.year, ti.number          /* le c fait
référence à la table country, le d pour disaster et le ti pour table_intermédiaire*/
FROM table_intermediaire ti
JOIN country c ON ti.country = c.name_country
JOIN disaster d ON ti.disaster = d.disaster;

```

```

DROP TABLE
CREATE TABLE
COPY 6448
INSERT 0 5
INSERT 0 17
INSERT 0 207
INSERT 0 6
INSERT 0 6448

```

Copie d'écran du résultat de la requête qui indique que tout s'est correctement déroulé.

| er  | country             | iso2 | iso3 | region_code | region   | sub_region_code | sub_region                      | disaster            | year | numb |
|-----|---------------------|------|------|-------------|----------|-----------------|---------------------------------|---------------------|------|------|
| --- | ---                 | ---  | ---  | ---         | ---      | ---             | ---                             | ---                 | ---  | ---  |
| 3   | Lebanon             | LB   | LBN  | 142         | Asia     | 145             | Western Asia                    | Storm               | 2015 |      |
| 1   | Nepal               | NP   | NPL  | 142         | Asia     | 34              | Southern Asia                   | Landslide           | 1989 |      |
| 4   | Japan               | JP   | JPN  | 142         | Asia     | 30              | Eastern Asia                    | Storm               | 1982 |      |
| 1   | Pakistan            | PK   | PAK  | 142         | Asia     | 34              | Southern Asia                   | Flood               | 1991 |      |
| 3   | Sri Lanka           | LK   | LKA  | 142         | Asia     | 34              | Southern Asia                   | Flood               | 2011 |      |
| 2   | Haiti               | HT   | HTI  | 19          | Americas | 419             | Latin America and the Caribbean | Flood               | 2006 |      |
| 1   | Kenya               | KE   | KEN  | 2           | Africa   | 202             | Sub-Saharan Africa              | Flood               | 1996 |      |
| 1   | Slovenia, Rep. of   | SI   | SVN  | 150         | Europe   | 39              | Southern Europe                 | Flood               | 2012 |      |
| 4   | Bangladesh          | BD   | BGD  | 142         | Asia     | 34              | Southern Asia                   | Storm               | 2015 |      |
| 1   | Pakistan            | PK   | PAK  | 142         | Asia     | 34              | Southern Asia                   | Extreme temperature | 2015 |      |
| 2   | Dominican Rep.      | DO   | DOM  | 19          | Americas | 419             | Latin America and the Caribbean | Flood               | 2009 |      |
| 1   | Sierra Leone        | SL   | SLE  | 2           | Africa   | 202             | Sub-Saharan Africa              | Flood               | 2004 |      |
| 2   | Netherlands, The    | NL   | NLD  | 150         | Europe   | 155             | Western Europe                  | Extreme temperature | 2005 |      |
| 1   | Australia           | AU   | AUS  | 9           | Oceania  | 53              | Australia and New Zealand       | Extreme temperature | 2009 |      |
| 1   | Nepal               | NP   | NPL  | 142         | Asia     | 34              | Southern Asia                   | Flood               | 2012 |      |
| 1   | Cambodia            | KH   | KHM  | 142         | Asia     | 35              | South-eastern Asia              | Drought             | 2005 |      |
| 2   | Madagascar, Rep. of | MG   | MDG  | 2           | Africa   | 202             | Sub-Saharan Africa              | Storm               | 2000 |      |
| 1   | India               | IN   | IND  | 142         | Asia     | 34              | Southern Asia                   | Extreme temperature | 2007 |      |

Copie d'écran d'une partie de la table obtenue avec le script ci-dessus.

## 2. Description commentée des différentes étapes du script de peuplement

Dans un premier temps, comme suggéré, j'ai créé une table temporaire avec les mêmes attributs que le fichier csv pour y stocker les tuples et pouvoir plus facilement attribuer à chaque attribut de la table temporaire une table.

Ensuite, j'ai copié les informations du fichier csv.zip que j'ai extrait pour pouvoir l'utiliser dans table\_intermediaire. J'ai précisé 'ENCODING 'UTF8'', car avant de le mettre lors de l'exécution du script, j'avais une erreur liée à l'encodage liée à certains caractères. Une fois copiée, j'ai vérifié que la table obtenue était correcte cohérente puis nous passons à la deuxième partie, c'est-à-dire le peuplement des tables que nous avons créées avec le premier script fait manuellement. J'ai utilisé à nouveau les recommandations de faire des projections avec 'INSERT INTO...SELECT'.

J'ai réalisé pour chaque attribut présent dans table\_intermediaire sa projection pour remplir les tables dans un ordre spécial ; en effet, il faut d'abord créer 'region' car des clés étrangères issues de cette table sont utilisées dans la plupart du reste des tables.

Pour la table 'climate\_disaster', j'ai réalisé deux jointures entre la table country et l'attribut country de TI (table\_intermediaire) pour associer le pays à sa catastrophe naturelle puis entre le nom du désastre dans la table 'disaster' et l'attribut disaster de TI pour associer le nom de l'événement au code de sa catastrophe. Ainsi, nous obtenons les tables qui avaient été demandées et peuplées avec le même nombre de lignes comme dans la figure 1.