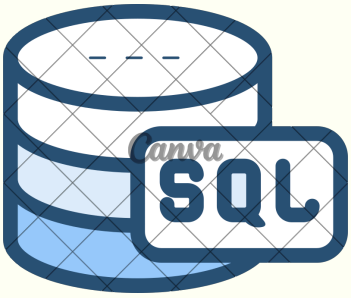
Ramasamy
Sanjai
Lerne

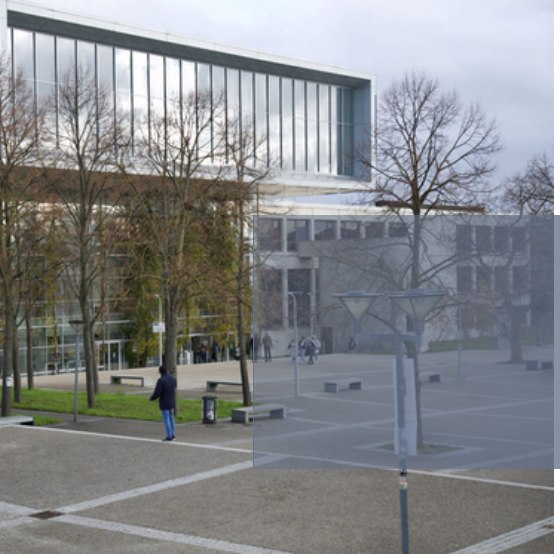


SAE :
BASE DE DONNÉE

PostgreSQL

IUT Villetaneuse
Université Sorbonne Paris Nord

2025



Sommaire

Université Sorbonne Paris Nord

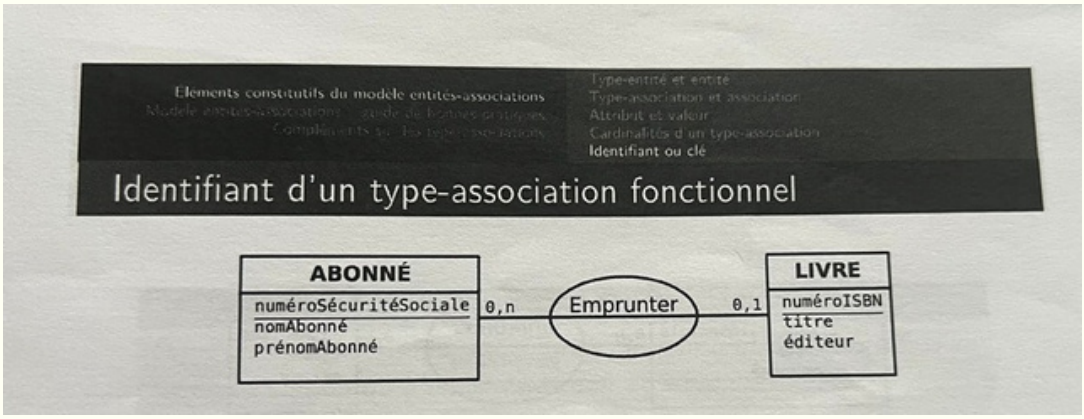
2.1 SCRIPT MANUEL DE CRÉATION DE LA BASE DE DONNÉE

1. Script SQL de création des tables

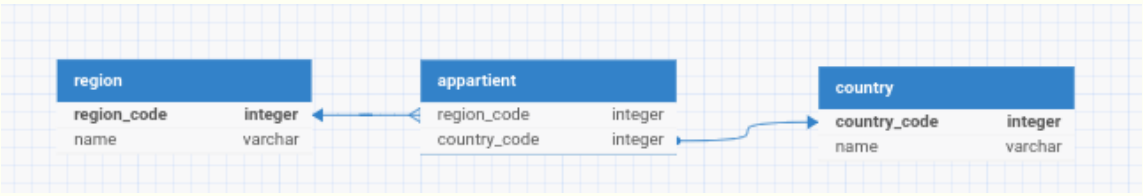
```
CREATE TABLE region (
  region_code INTEGER PRIMARY KEY,
  name VARCHAR NOT NULL);

CREATE TABLE sub_region (
  sub_region_code INTEGER PRIMARY KEY,
  name VARCHAR NOT NULL,
  region_code INTEGER REFERENCES region(region_code));

CREATE TABLE country (
  country_code INTEGER PRIMARY KEY,
  name VARCHAR NOT NULL,
  ISO2 CHAR(2),
  ISO3 CHAR(3),
  sub_region_code INTEGER REFERENCES sub_region(sub_region_code));

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,
  number INTEGER,
  PRIMARY KEY (country_code, disaster_code, year));
```

2.2 MODÉLISATION ET SCRIPT DE CRÉATION "AVEC AGL"

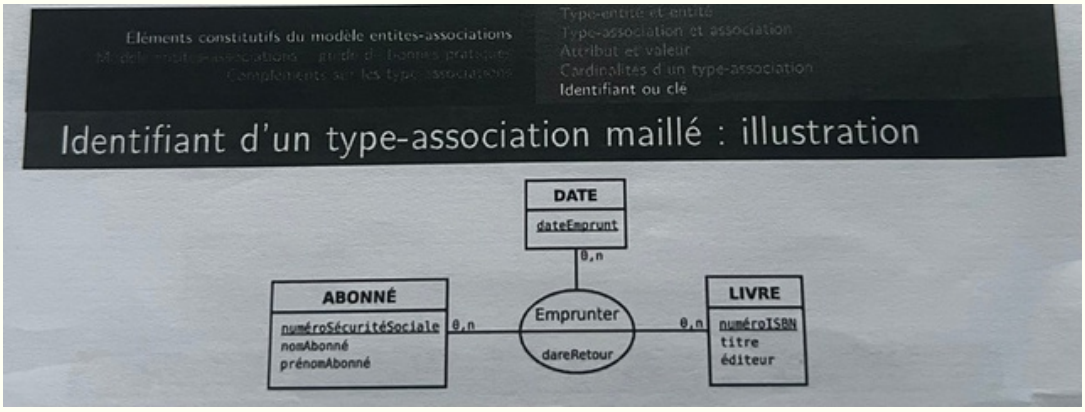
1. Illustration comparatives cours/AGL commentée d'une association fonctionnelle

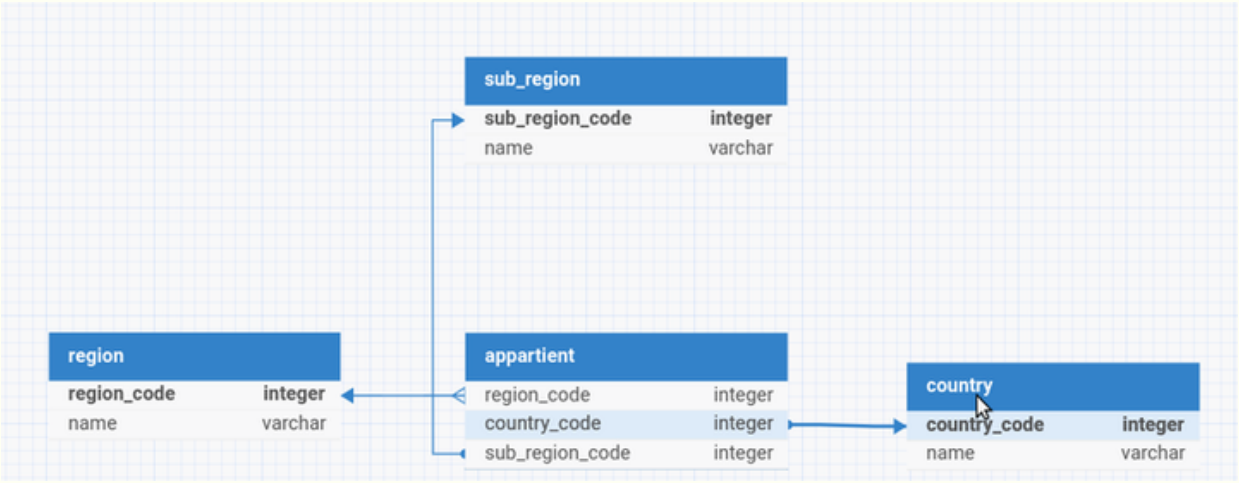


Exemple d'association fonctionne du cours

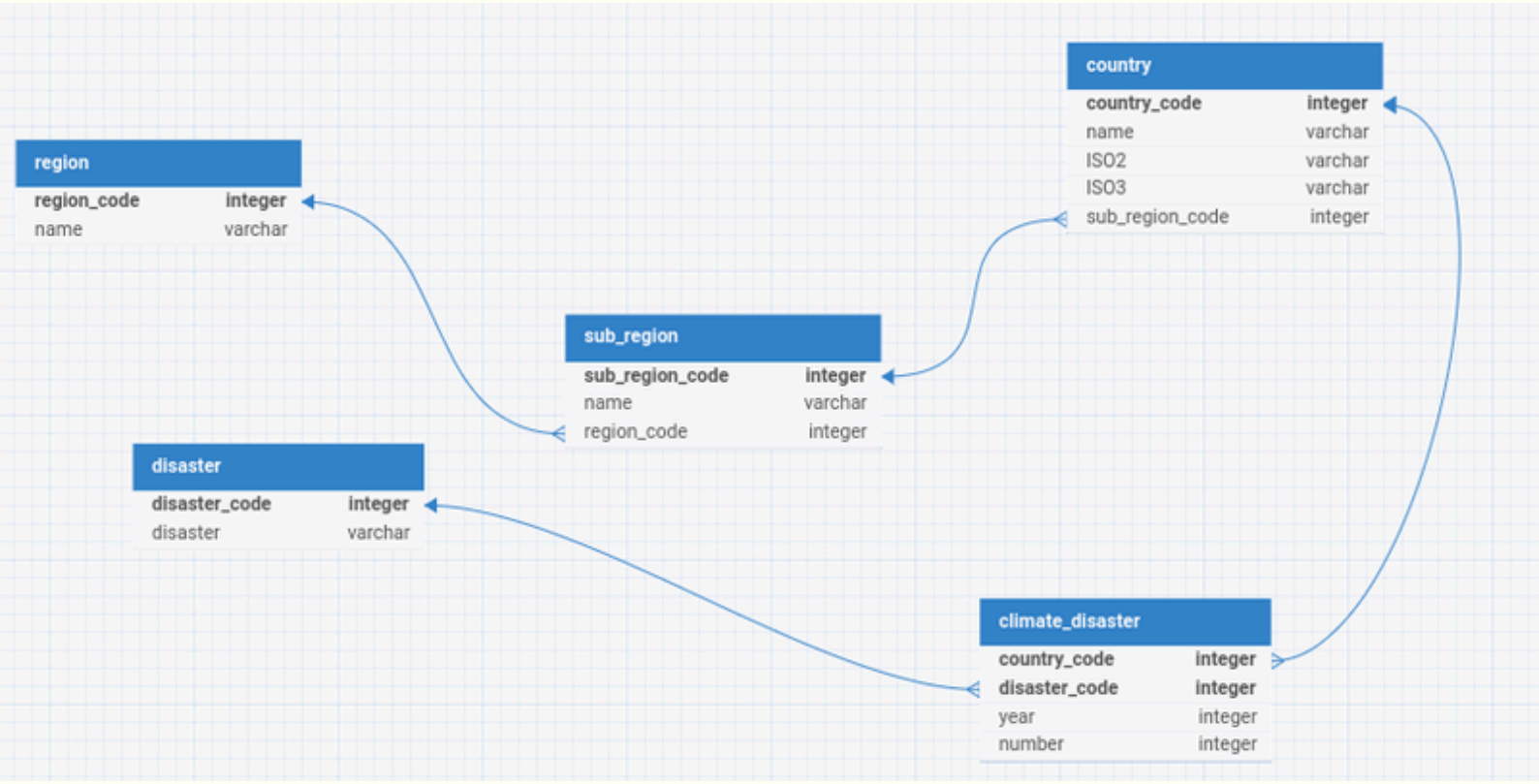


1. Illustration comparatives cours/AGL commentée d'une association maillet





3. Modèle physique de donnée réalisé avec l'AGL.



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

```
region {
 region_code integer pk increments unique
 name varchar
}

sub_region {
 sub_region_code integer pk increments unique
 name varchar
 region_code integer unique *> region.region_code
}

country {
 country_code integer pk increments unique
 name varchar
 ISO2 varchar
 ISO3 varchar
 sub_region_code integer *>
sub_region.sub_region_code
}

disaster {
 disaster_code integer pk increments
unique
 disaster varchar
}
climate_disaster {
 country_code integer pk *>
country.country_code
 disaster_code integer pk *>
disaster.disaster_code
 year integer
 number integer
}
```

5. Discussion sur les différences entre les scrpits produit manuellement et automatiquement

Le script manuel utilise une un modèle SQL traditionnelle, qui détaille chaque élément avec des mots script comme **CREATE TABLE, PRIMARY KEY ou REFERENCES,** il est beaucoup plus compréhensible et clair à comprendre. Mais par contre, le script généré automatiquement par l'AGL utilise une notation qui est très différent et simplfié propre à son outil de modélisation. Les notations de l'AGL sont par exemples : **pk pour clé primaire et des flèches *> pour les clés étrangères** qui est plus simple à comprendre dans un schéma global. Donc pour que le script soit utilisable dans une base de donnée, on devra effectuer une conversion.

2.3 PEUPLEMENT DES TABLES

1. Script de peuplement de la base de données & 2. Description commentée des différente étape du script

J'ai choisi la figure 1

1er - étape on va crée une table temporaire pour copier le fichier csv dans la table :

```
CREATE TEMP TABLE temp sanjai (
    country VARCHAR,
    iso2 CHAR(2),
    iso3 CHAR(3),
    region_code INTEGER,
    region VARCHAR,
    sub_region_code INTEGER,
    sub_region VARCHAR,
    disaster VARCHAR,
    year INTEGER,
    number INTEGER);
```

Cela donc permet de stocker les données du fichier csv avant qu'on les mette dans la table principale

La table temporaire s'appelle sanjai

2è étape - on va copier le contenu du fichier cvs dans la table qu'on a crée, c'est-à-dire la table temporaire "Sanjai"

```
\copy sanjai FROM 'Documents/Climate_related_disasters_frequency.csv'
DELIMITER ',' CSV HEADER;
```

Le csv Header permet d'indiquer au script d'ignorer la première ligne des noms des colonnes pour éviter qu'il consière cela comme une donnée

3è étape - On insére les peuplements dans chaque tables principale à partir du table temporaire qu'on mis précédement :

```
-- Region

INSERT INTO region (region_code, name)
SELECT DISTINCT region_code, region
FROM sanjai;

-- sub_region
INSERT INTO sub_region (sub_region_code, name, region_code)
SELECT DISTINCT sub_region_code, sub_region, region_code
FROM sanjai;

-- country

INSERT INTO country (name, ISO2, ISO3, sub_region_code)
SELECT DISTINCT country, ISO2, ISO3, sub_region_code
FROM sanjai;

-- disaster

INSERT INTO disaster(disaster)
SELECT DISTINCT disaster
FROM sanjai;

-- Insérer les données de désastres climatiques dans la table climate_disaster

INSERT INTO climate_disaster (country_code, disaster_code, year, number)
SELECT
 country.country_code,
 disaster.disaster_code,
 chiffre.year,
 chiffre.number

FROM sanjai chiffre
JOIN country country ON chiffre.country = country.name
JOIN disaster disaster ON chiffre.disaster = disaster.disaster;
```

4é étape - On pourrait vérifier le résultat :

```
SELECT * FROM region;
SELECT * FROM sub_region;
SELECT * FROM country;
SELECT * FROM disaster;
SELECT * FROM climate_disaster;
```

Maintenant qu'on a vérifier le resultat on supprime la table temporaire car on a insérer tous les donnée du fichier CSV dans les tables principales

```
DROP TABLE sanjai;
```

Le script complet :

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

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

CREATE TABLE sub_region (
   sub_region_code INTEGER PRIMARY KEY,
   name VARCHAR NOT NULL,
   region_code INTEGER REFERENCES region(region_code));

CREATE TABLE country (
   country_code SERIAL PRIMARY KEY,
   name VARCHAR NOT NULL,
   ISO2 CHAR(2),
   ISO3 CHAR(3),
   sub_region_code INTEGER REFERENCES sub_region(sub_region_code));

CREATE TABLE disaster (
   disaster_code SERIAL 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,
   number INTEGER,
   PRIMARY KEY (country_code, disaster_code, year));
```

```
CREATE TEMP TABLE sanjai (
   country VARCHAR,
   iso2 CHAR(2),
   iso3 CHAR(3),
   region_code INTEGER,
   region VARCHAR,
   sub_region_code INTEGER,
   sub_region VARCHAR,
   disaster VARCHAR,
   year INTEGER,
   number INTEGER);

\copy sanjai FROM 'Documents/Climate_related_disasters_frequency.csv' CSV HEADER;

-- région
INSERT INTO region (region_code, name)
SELECT DISTINCT region_code, region
FROM sanjai;

-- sub_region
INSERT INTO sub_region (sub_region_code, name, region_code)
SELECT DISTINCT sub_region_code, sub_region, region_code
FROM sanjai;

-- country
INSERT INTO country (name, ISO2, ISO3, sub_region_code)
SELECT DISTINCT country, ISO2, ISO3, sub_region_code
FROM sanjai;

-- disaster
INSERT INTO disaster(disaster)
SELECT DISTINCT disaster
FROM sanjai;

-- climate_disaster
INSERT INTO climate_disaster (country_code, disaster_code, year, number)

SELECT
 country.country_code,
 disaster.disaster_code,
 chiffre.year,
 chiffre.number

FROM sanjai chiffre
JOIN country country ON chiffre.country = country.name
JOIN disaster disaster ON chiffre.disaster = disaster.disaster;

SELECT * FROM region;
SELECT * FROM sub_region;
SELECT * FROM country;
SELECT * FROM disaster;
SELECT * FROM climate_disaster;
```

Voici donc un aperçu du resultat finale

Terminal - sanjai@sanjai-Latitude-E7470: ~

File Edit View Terminal Tabs Help

country_code	name	iso2	iso3	sub_region_code
1	Montserrat	MS	MSR	419
2	Rwanda	RW	RWA	202
3	Sudan	SD	SDN	15
4	Lebanon	LB	LBN	145
5	Bahamas, The	BS	BHS	419
6	Guyana	GY	GUY	419
7	Congo, Rep. of	CG	COG	202
8	Bermuda	BM	BMU	21
9	Benin	BJ	BEN	202
10	Samoa	WS	WSM	61
11	Philippines	PH	PHL	35
12	New Zealand	NZ	NZL	53
13	Tajikistan, Rep. of	TJ	TJK	143
14	Honduras	HN	HND	419
15	Sri Lanka	LK	LKA	34
16	Austria	AT	AUT	155
17	Mauritania, Islamic Rep. of	MR	MRT	202

File Edit View Terminal Tabs Help

country_code	disaster_code	year	number
139	1	1998	1
155	1	2017	1
67	1	2014	1
190	1	2005	1
59	1	1996	1
152	1	1981	1
190	1	2011	1
190	1	2010	1
99	1	2012	1
99	1	2017	1
67	1	2012	1
110	1	1981	1
67	1	2009	1
67	1	2001	1
33	1	2012	1
56	1	1999	1
195	1	1999	1

region_code	name
2	Africa
142	Asia
9	Oceania
19	Americas
150	Europe

(5 rows)

sub_region_code	name	region_code
57	Micronesia	9
154	Northern Europe	150
15	Northern Africa	2
39	Southern Europe	150
145	Western Asia	142
35	South-eastern Asia	142
202	Sub-Saharan Africa	2
54	Melanesia	9
34	Southern Asia	142
53	Australia and New Zealand	9
30	Eastern Asia	142
21	Northern America	19
143	Central Asia	142
61	Polynesia	9
151	Eastern Europe	150
155	Western Europe	150
419	Latin America and the Caribbean	19

(17 rows)

disaster_code	disaster
1	Drought
2	Extreme temperature
3	Wildfire
4	Flood
5	Landslide
6	Storm

(6 rows)