IUT Villetaneuse
Université Sorbonne Paris Nord

UNIVERSITÉ SORBONNE PARIS NORD

Boran CAV
GROUPE: LERNE
Etudiant en 1ère année de BUT Informatique

SAE:
BASE DE DONNÉES et LANGAGE SQL

SOMMAIRE

I - Script manuel de création de la base de données

Voici le code SQL permettant la création des tables:

```sql
CREATE TABLE region (
    region_code VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);
CREATE TABLE sub_region (
    sub_region_code VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    region_code VARCHAR(10) NOT NULL,
    FOREIGN KEY (region_code) REFERENCES region(region_code)
);
CREATE TABLE country (
    country_code VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    ISO2 CHAR(2) NOT NULL,
    ISO3 CHAR(3) NOT NULL,
    sub_region_code VARCHAR(10) NOT NULL,
    FOREIGN KEY (sub_region_code) REFERENCES sub_region(sub_region_code)
);
CREATE TABLE disaster (
    disaster_code VARCHAR(10) PRIMARY KEY,
    disaster VARCHAR(50) NOT NULL
);
CREATE TABLE climate_disaster (
    country_code VARCHAR(10) NOT NULL,
    disaster_code VARCHAR(10) NOT NULL,
    year INT NOT NULL,
    number INT NOT NULL,
    PRIMARY KEY (country_code, disaster_code, year),
    FOREIGN KEY (country_code) REFERENCES country(country_code),
    FOREIGN KEY (disaster_code) REFERENCES disaster(disaster_code)
);
```

II - Modélisation et script de création avec AGL

A - Illustrations comparatives cours/AGL commenté d'une association fonctionnelle

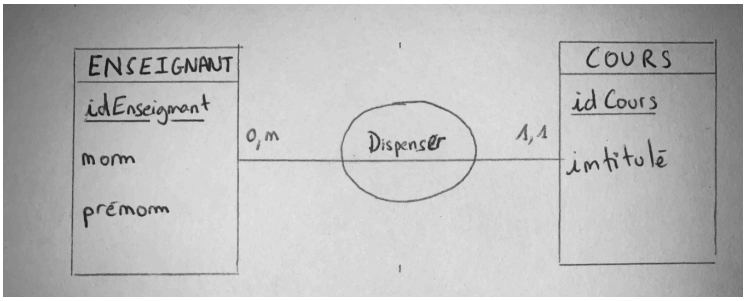

Figure II.A.1 : Illustrations comparatives cours d'une association fonctionnelle

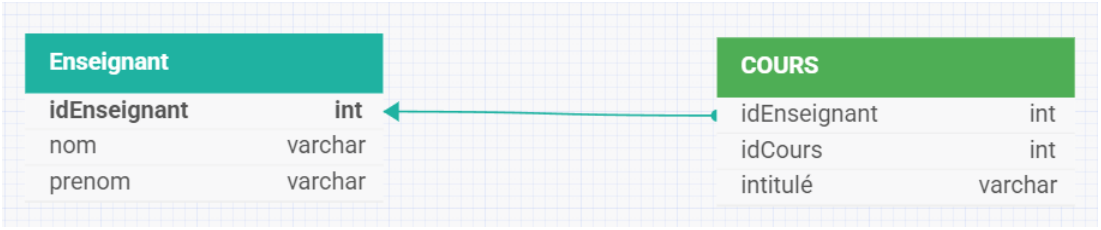

Figure II.A.2 : Illustrations comparatives AGL d'une association fonctionnelle

Commentaire:

Dans le modèle AGL sur **DB Designer**, cette relation fonctionnelle n'a pas de table intermédiaire **Dispenser**. Au lieu de cela, la table **Cours** contient une clé étrangère `idEnseignant`, ce qui établit une relation directe avec la table **Enseignant** ce qui respecte bien la cardinalité du modèle entités-association.

B - Illustrations comparatives cours/AGL commenté d'une association maillé

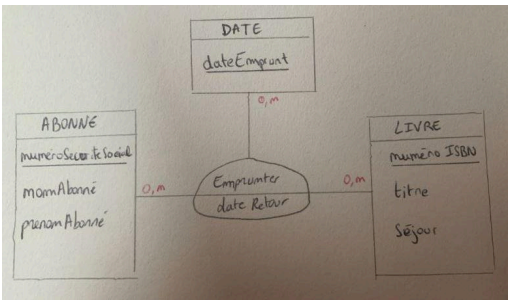

Figure II.B.1 : Illustrations comparatives cours d'une association maillé

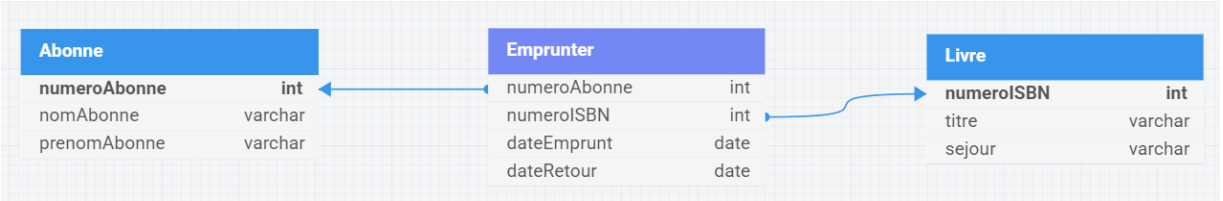

Figure II.B.2 : Illustrations comparatives AGL d'une association maillé

Commentaire:

Dans le modèle AGL sur **DB Designer**, cette relation maillée est modélisée à l'aide de la table **Emprunter**, qui joue le rôle d'une table d'association et contient les clés étrangères numeroAbonne, numeroISBN et dateEmprunt ce qui établit les liens avec les tables **Abonne**, **Livre** et les dates d'emprunt.

C - Modèle physique de données réalisé avec L'AGL

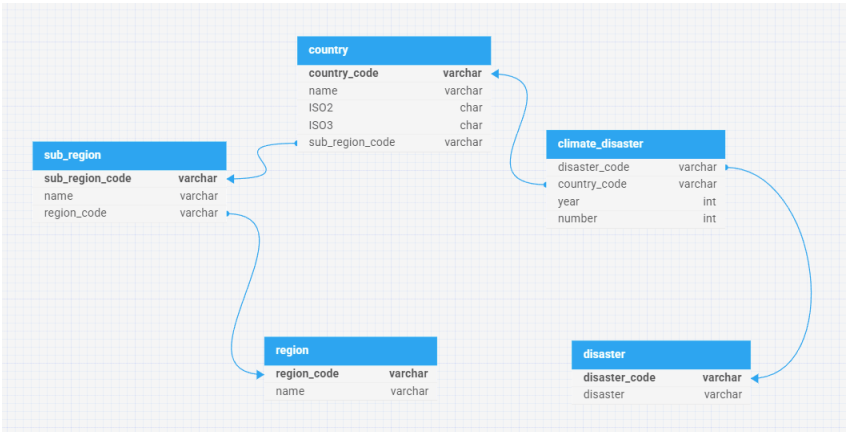

Figure II.C: Modèle physique de données réalisé avec L'AGL

D - Script SQL de création des tables généré automatique par l'AGL

```
CREATE TABLE IF NOT EXISTS "region" (
	"region_code" varchar(10) NOT NULL,
	"name" varchar(100) NOT NULL,
	PRIMARY KEY ("region_code")
);

CREATE TABLE IF NOT EXISTS "sub_region" (
	"sub_region_code" varchar(10) NOT NULL,
	"name" varchar(100) NOT NULL,
	"region_code" varchar(10) NOT NULL,
	PRIMARY KEY ("sub_region_code")
);

CREATE TABLE IF NOT EXISTS "country" (
	"country_code" varchar(10) NOT NULL,
	"name" varchar(100) NOT NULL,
	"ISO2" varchar(2) NOT NULL,
	"ISO3" varchar(3) NOT NULL,
	"sub_region_code" varchar(10) NOT NULL,
	PRIMARY KEY ("country_code")
);

CREATE TABLE IF NOT EXISTS "disaster" (
	"disaster_code" varchar(10) NOT NULL,
	"disaster" varchar(50) NOT NULL,
	PRIMARY KEY ("disaster_code")
);

CREATE TABLE IF NOT EXISTS "climate_disaster" (
	"disaster_code" varchar(10) NOT NULL,
	"country_code" varchar(10) NOT NULL,
	"year" bigint NOT NULL,
	"number" bigint NOT NULL
);


ALTER TABLE "sub_region" ADD CONSTRAINT "sub_region_fk2" FOREIGN KEY ("region_code") REFERENCES "region"("region_code");
ALTER TABLE "country" ADD CONSTRAINT "country_fk4" FOREIGN KEY ("sub_region_code") REFERENCES "sub_region"("sub_region_code");

ALTER TABLE "climate_disaster" ADD CONSTRAINT "climate_disaster_fk0" FOREIGN KEY ("disaster_code") REFERENCES "disaster"("disaster_code");

ALTER TABLE "climate_disaster" ADD CONSTRAINT "climate_disaster_fk1" FOREIGN KEY ("country_code") REFERENCES "country"("country_code");
```

E - Discussion sur les différences entre les scripts produits manuellement et automatique

Dans le script manuelle, on peut lire que le nom des tables est sans guillemets tandis que DB Designer met automatiquement des guillemets aux noms de ces tables. Les guillemets doubles permettent de garder le nom avec ces majuscules tandis que sur postgres l'absence de guillemets met le mot en minuscules directement.

On peut noter également que les clés étrangères sont définies directement lors de la création des tables avec la commande FOREIGN KEY dans mon script manuel.
Tandis que dans le script fait par **DB Designer**, les clés étrangères sont ajoutées **après la création des tables** avec la commande ALTER TABLE.
On remarque également que le script généré par DB Designer utilise bigint de int
car bigint peut stocker des valeurs beaucoup grandes parce qu'elle est utilise 8 octet de mémoire contre 4 octet pour int.

III - Peuplement des tables

On crée d'abord notre table temporaire temp_climate_data à l'aide de CREATE TEMP TABLE, cette table temporaire va contenir toute les colonnes du fichier csv :

```
CREATE TEMP TABLE temp_climate_data (
    country VARCHAR(100),
    iso2 CHAR(2),
    iso3 CHAR(3),
    region_code VARCHAR(10),
    region VARCHAR(100),
    sub_region_code VARCHAR(10),
    sub_region VARCHAR(100),
    disaster VARCHAR(50),
    year INT,
    number INT
);
```

Ensuite grâce à la commande \copy ,on va copier dans notre table temporaire les valeurs de notre fichier csv en précisant le délimiteur qui est une virgule.

```
\copy temp_climate_data FROM 'Climate_related_disasters_frequency.csv' DELIMITER
',' CSV HEADER;
```

On vient maintenant peupler les différentes tables avec la commande `INSERT INTO` qui nous permet d'insérer les valeurs de la table temporaire aux tables. On a peuplé ci-dessous la table `region` :

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

Mais également la table `sub_region` :

```
INSERT INTO sub_region (sub_region_code, name, region_code)
SELECT DISTINCT sub_region_code, sub_region, region_code
FROM temp_climate_data;
```

La table `country` :

```
INSERT INTO country (country_code, name, ISO2, ISO3, sub_region_code)
SELECT DISTINCT iso3, country, iso2, iso3, sub_region_code
FROM temp_climate_data;
```

La table `disaster`:

```
INSERT INTO disaster (disaster_code, disaster)
SELECT DISTINCT disaster AS disaster_code, disaster
FROM temp_climate_data;
```

La table `climate_disaster`:

```
INSERT INTO climate_disaster (country_code, disaster_code, year, number)
SELECT
    iso3 AS country_code,
    AS disaster_code,year,number
FROM temp_climate_data;
```