Nom : Zasir MOINOOTHEEN
Numéro étudiant : 12300620
Groupe : Euros

Rapport synthétique

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

Université Sorbonne Paris Nord

Sommaire

Avant Propos

Ce rapport présente le résultat de ma SAÉ de conception et de réalisation d'une base de données relationnelle. Le sujet de ce projet était de modéliser et de peupler une base de données sur le niveau de liberté des pays du monde au cours des années, en utilisant des données issues d'un fichier csv.
Le but de ce projet était de mettre en pratique les notions de modélisation entité/association, de schéma relationnel, de normalisation, de requêtes SQL et de peuplement de tables.

Pour réaliser ce projet, j'ai utilisé le système de gestion de base de données PostgreSQL, qui était un choix imposé. J'ai également utilisé l'outil MySQL Workbench, qui est un Atelier de Génie Logiciel (AGL), qui permet de créer et de modifier des modèles de données, d'exécuter des requêtes SQL et de gérer des serveurs de base de données.

Au cours de ce projet, j'ai rencontré plusieurs difficultés, notamment liées aux problèmes de type, de chemin, de dépendance, d'insertion et de jointure. J'ai réussi à les surmonter en faisant des recherches sur internet, en consultant des sources fiables et en testant différentes solutions. Je vais détailler ces difficultés et les solutions que j'ai trouvées dans la suite de ce rapport.

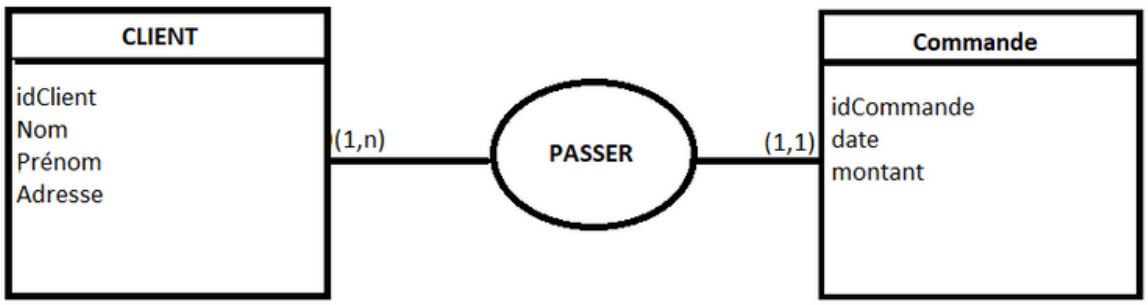
I. Script manuel de création de la base de données

Voici le script réalisé à la main pour créer les tables :

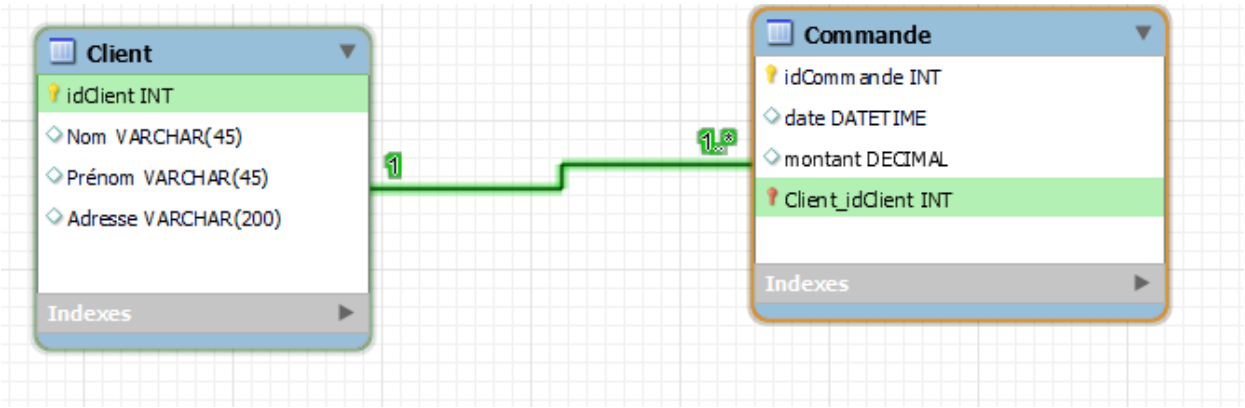
```
CREATE TABLE region(
 region_code INT PRIMARY KEY,
 name VARCHAR(50)`
);

CREATE TABLE COUNTRY (
  id_country SERIAL PRIMARY KEY,
  name VARCHAR(255),
  is_ldc BOOLEAN,
  region_code INT REFERENCES REGION(region_code)
);

CREATE TABLE STATUS (
  status VARCHAR(2) PRIMARY KEY
);

CREATE TABLE FREEDOM (
  id_country INT REFERENCES COUNTRY(id_country),
  year INT,
  civil_liberties INT,
  political_rights INT,
  status VARCHAR(2) REFERENCES STATUS(status),
  PRIMARY KEY (id_country, year)
);
```

II. Modélisation et script de création avec l'AGL

1. Association Fonctionnelle



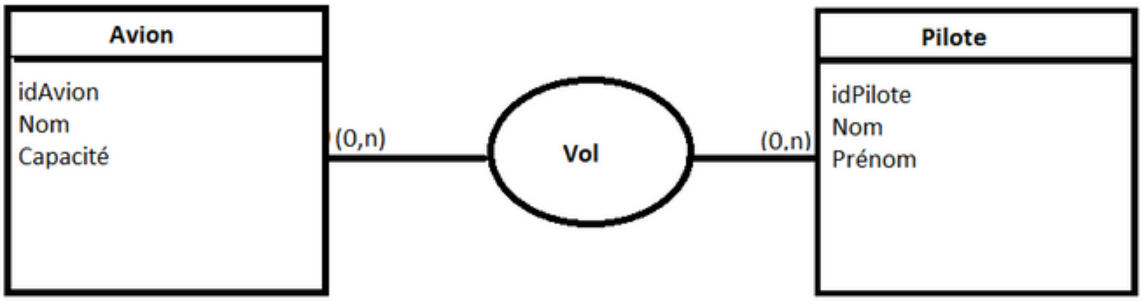
Modèle Conceptuel de Données (MCD)



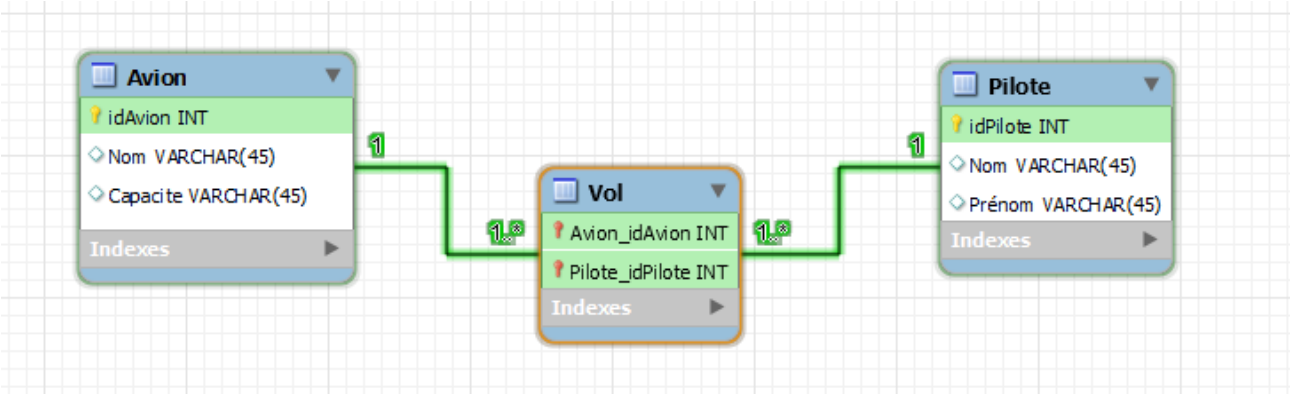
Modèle Physique de Données (MPD)

Cette association fonctionnelle représente le fait qu'un client puisse passer une ou plusieurs commandes, tandis que une commande ne peut être effectuée que par un seul client.
Nous pouvons également constater que lorsque l'on passe du MCD au MPD, une clé étrangère qui fait référence à la clé primaire de la table Client se crée du côté de la table dépendante, c'est-à-dire la table Commande.

2. Association Maillée



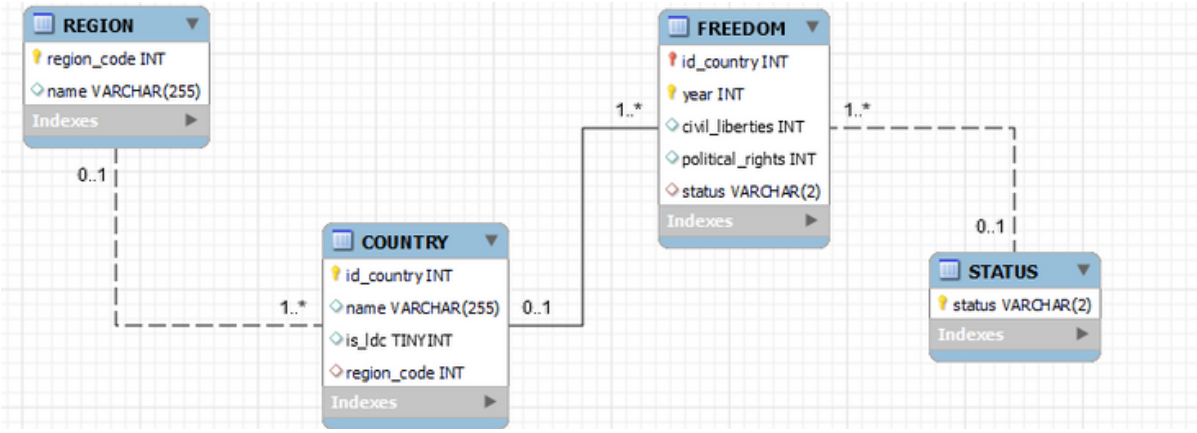
Modèle Conceptuel de Données (MCD)



Modèle Physique de Données (MPD)

Cette relation représente une association maillé, qui représente le fait qu'un avion et un pilote peuvent effectuer un seul vol, alors que chaque avion peut participer à plusieurs vols et chaque pilote peut également participer à plusieurs vols. Nous pouvons également constater que lorsque l'on relie les deux tables, une table associative se crée (ici la table Vol), avec les clés primaires des deux premières tables.

3. Modèle Physique de Données



Modèle MCD de la figure 2 : Modélisation entités-associations de la base de données

4. Script SQL des tables, générés par l'AGL

Voici le script généré par l'AGL, ici Mysql Workbench en l'occurence :

```
-- MySQL Script generated by MySQL Workbench
-- Mon Jan 22 22:47:36 2024
-- Model: New Model  Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZE
RO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `mydb` ;

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
SHOW WARNINGS;
```

```
USE `mydb` ;

-- -----------------------------------------------------
-- Table `COUNTRY`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `COUNTRY` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `COUNTRY` (
 `id_country` INT NULL DEFAULT NULL,
 `name` VARCHAR(255) NULL DEFAULT NULL,
 `is_ldc` TINYINT NULL DEFAULT NULL,
 `region_code` INT NULL DEFAULT NULL,
 PRIMARY KEY (`id_country`),
 CONSTRAINT ``
 FOREIGN KEY (`region_code`)
 REFERENCES `REGION` (`region_code`));

SHOW WARNINGS;
CREATE INDEX ON `COUNTRY` (`region_code` ASC) VISIBLE;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `FREEDOM`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `FREEDOM` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `FREEDOM` (
 `id_country` INT NULL DEFAULT NULL,
 `year` INT NULL DEFAULT NULL,
 `civil_liberties` INT NULL DEFAULT NULL,
 `political_rights` INT NULL DEFAULT NULL,
 `status` VARCHAR(2) NULL DEFAULT NULL,
 PRIMARY KEY (`id_country`, `year`),
 CONSTRAINT ``
 FOREIGN KEY (`id_country`)
 REFERENCES `COUNTRY` (`id_country`),
 CONSTRAINT ``
 FOREIGN KEY (`status`)
 REFERENCES `STATUS` (`status`));
```

```
USHOW WARNINGS;
CREATE INDEX ON `FREEDOM` (`status` ASC) VISIBLE;

SHOW WARNINGS;
CREATE INDEX ON `FREEDOM` (`status` ASC) VISIBLE;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `REGION`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `REGION` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `REGION` (
 `region_code` INT NULL DEFAULT NULL,
 `name` VARCHAR(255) NULL DEFAULT NULL,
 PRIMARY KEY (`region_code`));

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `STATUS`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `STATUS` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `STATUS` (
 `status` VARCHAR(2) NULL DEFAULT NULL,
 PRIMARY KEY (`status`));

SHOW WARNINGS;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

5. Différences scripts manuels / automatiques

Nous pouvons voir que le script généré par l'AGL est plus complexe et détaillé.

Il comprend des suppression de schémas et de table afin d'éviter des erreurs.
Il indique également des contraintes de clé étrangère, par exemple la table *REGION* a une clé étrangère qui fait référence à la table *COUNTRY*.
De plus, le premier script utilise l'instruction *SHOW WARNINGS;* après chaque instruction *CREATE TABLE* et *DROP TABLE*, afin d'indiquer des éventuels erreurs.
Mais aussi, il utilise des instructions spécifiques à MySQL telles que *SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0*; qui sont utilisées pour modifier temporairement certains paramètres de la base de données.

Pour conclure, les deux scripts créent bien les mêmes tables dans une base de données, mais le premier script est plus complexe et détaillé, et gère également les éventuels erreurs afin d'optimiser les performances, tandis que le script fait manuellement, est beaucoup plus simple et fonctionne sur différents SGBD qui supportent le langage SQL standard.

III. Peuplement des tables

Pour peupler les tables, voici comment j'ai procédé :

- Premièrement, j'ai recréé les tables de manière plus propre, en précisant notamment les clés étrangères :

```
DROP TABLE IF EXISTS COUNTRY CASCADE;
DROP TABLE IF EXISTS FREEDOM CASCADE;
DROP TABLE IF EXISTS REGION CASCADE;
DROP TABLE IF EXISTS STATUS CASCADE;

CREATE TABLE REGION (
  region_code INT PRIMARY KEY,
  name VARCHAR(255)
);


CREATE TABLE COUNTRY (
  id_country SERIAL PRIMARY KEY,
  name VARCHAR(255),
  is_ldc BOOLEAN,
  region_code INT REFERENCES REGION(region_code)
);


CREATE TABLE STATUS (
  status VARCHAR(2) PRIMARY KEY
);

CREATE TABLE FREEDOM (
  id_country INT REFERENCES COUNTRY(id_country),
  year INT,
  civil_liberties INT,
  political_rights INT,
  status VARCHAR(2) REFERENCES STATUS(status),
  PRIMARY KEY (id_country, year)
);
```

- Ensuite, j'ai créé une table temporaire qui stockera toutes les données du fichier csv grâce à l'instruction \copy :

```
DROP TABLE IF EXISTS TableTemp;

CREATE TABLE IF NOT EXISTS TableTemp (
 country VARCHAR(100),
 year INT,
 CL INT,
 PR INT,
 status VARCHAR(2),
 region_code INT,
 region_name VARCHAR(75),
 is_ldc BOOLEAN) ;

\copy TableTemp FROM 'D:/Cours/BUT INFO1/SAE S105 (BDD)/freedom.csv' WITH
(FORMAT csv, HEADER true, DELIMITER ',');
```

- Et enfin, j'ai projeté les informations de la table temporaire, dans les différentes tables, puis j'ai supprimé la table temporaire car elle ne faisait pas partie de la base de données.

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

INSERT INTO country (name, is_ldc, region_code)
SELECT DISTINCT country, is_ldc, region_code
FROM TableTemp;

INSERT INTO status (status)
SELECT DISTINCT status
FROM TableTemp;

INSERT INTO freedom (id_country, year, civil_liberties, political_rights, status)
SELECT c.id_country, t.year, t.CL, t.PR, t.status
FROM TableTemp t
JOIN country c ON t.country = c.name;

DROP TABLE TableTemp;
```

Résultat du code :

```
postgres=# \i 'D:/Cours/BUT INFO1/SAE S105 (BDD)/peuplement.sql'
DROP TABLE
CREATE TABLE
COPY 4979
INSERT 0 5
INSERT 0 193
INSERT 0 3
INSERT 0 4979
DROP TABLE
```

Après avoir décrit les étapes du processus de peuplement des tables, je vais maintenant présenter les difficultés que j'ai dû surmonter.

La première complication que j'ai rencontrée était liée aux problèmes de type. Lorsque j'ai créé la table temporaire pour stocker toutes les données, j'ai mal recopié les types. Par exemple, pour la colonne *is_ldc*, j'avais mis le type *INT*, alors que j'avais mis le type *BOOLEAN* dans la table country.

Ensuite, la deuxième complication que j'ai rencontrée concernait les problèmes de chemin avec la commande \copy. Sur Windows, le chemin vers un fichier est marqué par des backslash '\', alors que \copy ne prend pas les backslash. Il fallait donc que je change les backslash par des slash '/' pour que ça fonctionne.

J'ai également eu affaire à des problèmes de dépendance. J'avais créé des tables dans le mauvais ordre et du coup, quand je devais remplir la table country avec la colonne region_code, mais que la table region n'était pas définie, ça me mettait des erreurs.

Et enfin, une autre complication que j'ai pu rencontrer était due aux erreurs pour la commande INSERT INTO country. J'avais oublié de spécifier les colonnes à remplir. Par conséquent, la clé primaire id_country, qui est de type SERIAL, a pris la valeur du nom de la ville au lieu de s'incrémenter automatiquement. Pour résoudre ce problème, j'ai dû indiquer les colonnes correspondantes, comme ceci : INSERT INTO country(name,is_ldc,region_code).