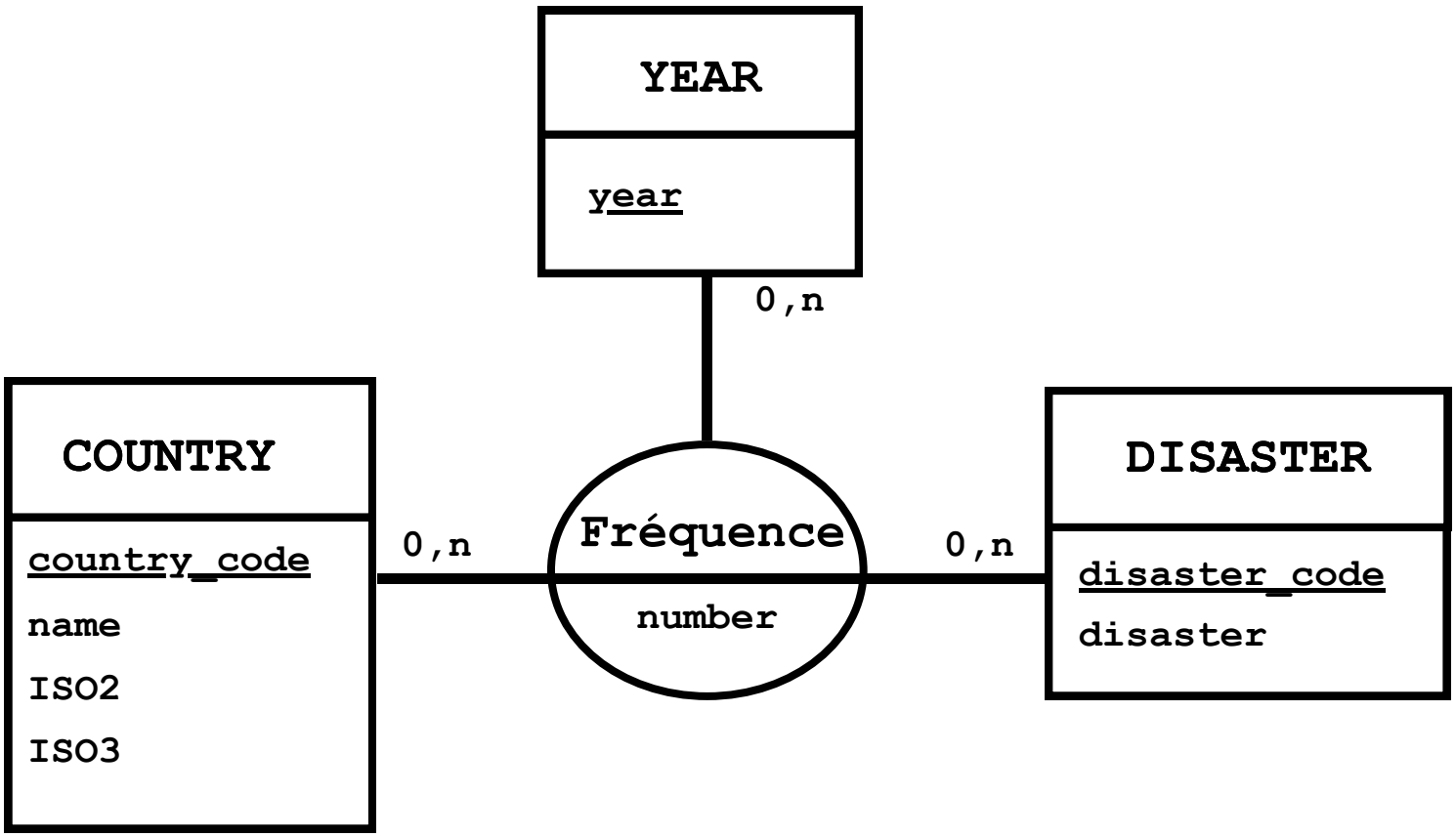
MASTOURI Ambrine
Groupe Erymanthe

BUT1 Informatique
2024-2025

SAE S104 : Création d'une base de données

Sommaire :


YEAR
year
0,n
COUNTRY
country_code
name
ISO2
ISO3
0,n
Fréquence
number
0,n
DISASTER
disaster_code
disaster

1- Script manuel de création de la base de données

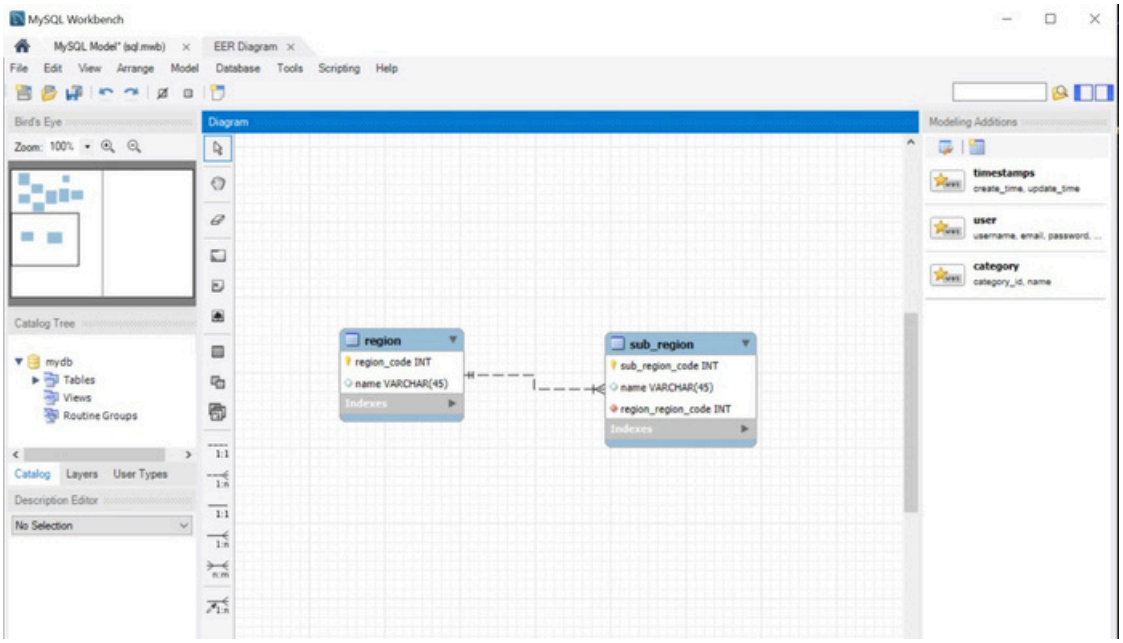
Voici le script manuel de création de la base de données, conformément à l'exemple donné sous forme de schéma relationnel :

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

CREATE TABLE sub_region (
sub_region_code INTEGER PRIMARY KEY,
name VARCHAR,
region_code INTEGER REFERENCES region (region_code)
) ; CREATE TABLE country (
country_code INTEGER PRIMARY KEY,
name VARCHAR,
ISO2 VARCHAR(2),
ISO3 VARCHAR(3),
sub_region_code INTEGER REFERENCES sub_region (sub_region_code)
) ; CREATE TABLE disaster (
disaster_code INTEGER PRIMARY KEY,
disaster VARCHAR) ;

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

2- Modélisation et script de création « avec AGL »

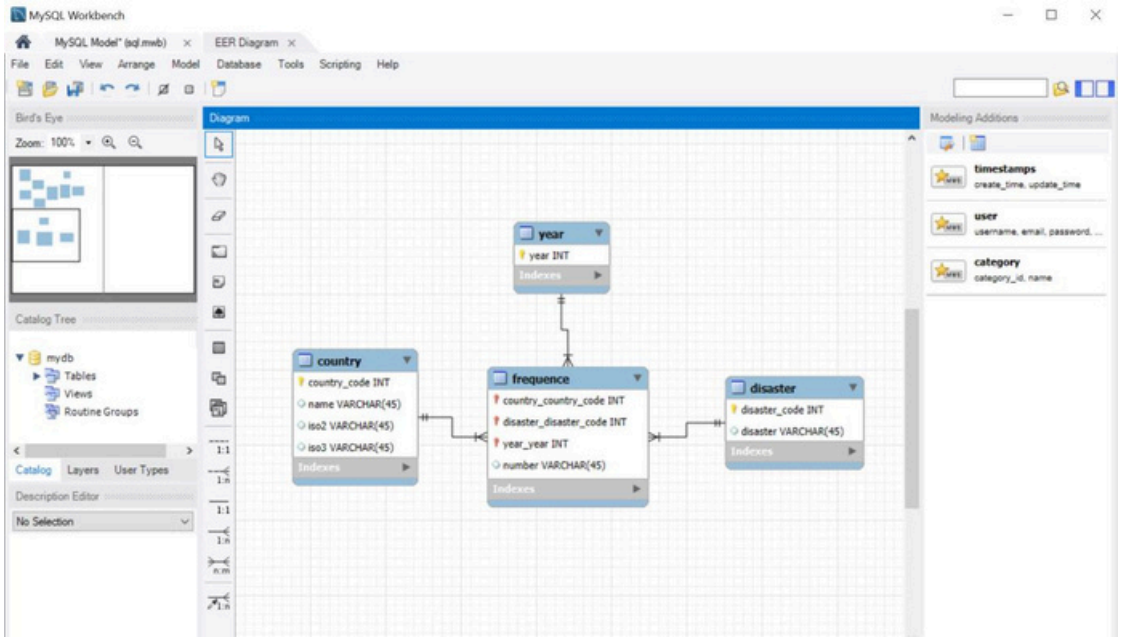
Pour cette partie, nous utiliserons le logiciel MySQL Workbench.

Pour commencer, voici la modélisation d'une association fonctionnelle sur ce logiciel, puis selon la méthode vue en cours :



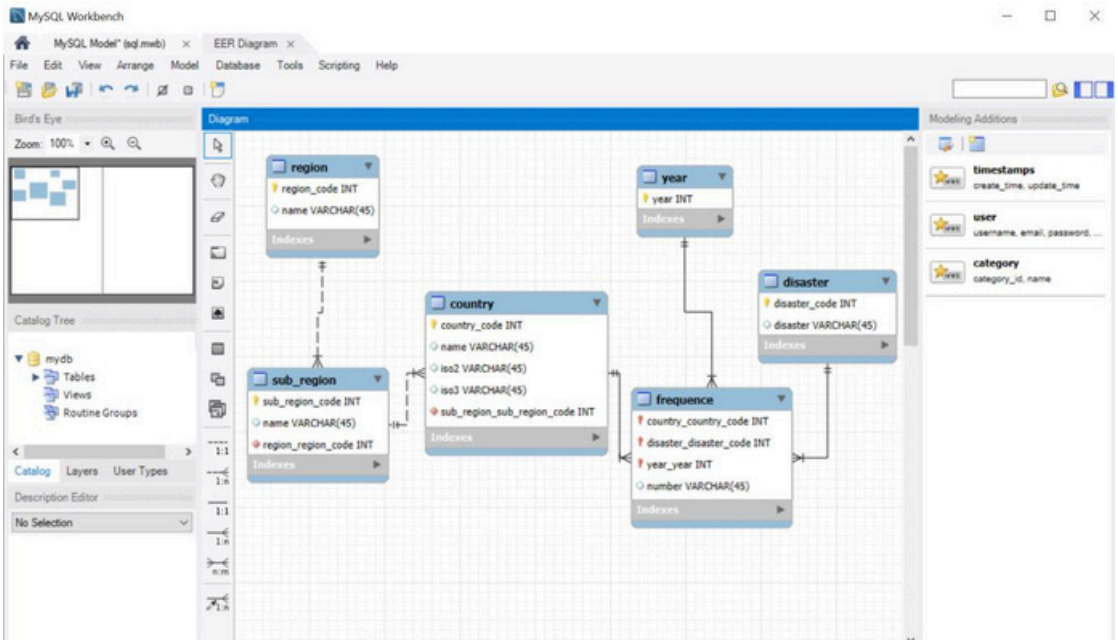
Ici, au lieu d'avoir une cardinalité et le groupe *Situer* comme dans la méthode vue en cours, il y a une flèche reliant les deux tables ainsi que l'ajout de la clé étrangère, précédée d'un symbole rouge afin de la différencier de la clé primaire, en jaune.

Ensuite, voici la modélisation d'une association maillée sur ce logiciel, puis selon la méthode vue en cours :



Ici, bien que la cardinalité ne soit toujours pas présente, les flèches convergent vers une même table correspondant au groupe *Fréquence*. Contrairement à la méthode vue en cours, *Fréquence* est une table à part entière qui a pour clés primaires les clés étrangères country_code, disaster_code et year, faisant respectivement référence aux tables country, disaster
et year.

Ensuite, voici le modèle physique de données correspondant au modèle conceptuel de données présenté dans le sujet de cette SAE :



Enfin, voici le script SQL de création de tables généré automatiquement par l'AGL :

```
-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`region`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`region` (

  `region_code` INT NOT NULL,

  `name` VARCHAR(45) NULL,

  PRIMARY KEY (`region_code`))

ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`sub_region`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`sub_region` (
  `sub_region_code` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `region_region_code` INT NOT NULL,
  PRIMARY KEY (`sub_region_code`),
  CONSTRAINT `fk_sub_region_region1`
    FOREIGN KEY (`region_region_code`)
    REFERENCES `mydb`.`region` (`region_code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`country`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`country` (
  `country_code`   INT   NOT   NULL,   `name`
  VARCHAR(45) NULL, `iso2` VARCHAR(45) NULL,
  `iso3`           VARCHAR(45)          NULL,
  `sub_region_sub_region_code` INT NOT NULL,
  PRIMARY  KEY  (`country_code`),  CONSTRAINT
  `fk_country_sub_region1`

    FOREIGN KEY (`sub_region_sub_region_code`)
    REFERENCES `mydb`.`sub_region` (`sub_region_code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`year`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`year` (
  `year` INT NOT NULL,
  PRIMARY KEY (`year`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`disaster`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`disaster` (
  `disaster_code` INT NOT NULL,
  `disaster` VARCHAR(45) NULL,
  PRIMARY KEY (`disaster_code`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`frequence`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`frequence` (
  `country_country_code` INT NOT NULL,
  `disaster_disaster_code` INT NOT NULL,
  `year_year` INT NOT NULL,
  `number` VARCHAR(45) NULL,
 PRIMARY KEY (`country_country_code`, `disaster_disaster_code`,
`year_year`),
  CONSTRAINT `fk_country_has_disaster_country2`
    FOREIGN KEY (`country_country_code`)
    REFERENCES `mydb`.`country` (`country_code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_country_has_disaster_disaster2`
```

```
    FOREIGN KEY (`disaster_disaster_code`)
    REFERENCES `mydb`.`disaster` (`disaster_code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_country_has_disaster_year1`
    FOREIGN KEY (`year_year`)
    REFERENCES `mydb`.`year` (`year`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Ce script SQL généré par MySQL Workbench comporte plusieurs différences avec celui que j'ai réalisé précédemment :

Pour commencer, le script généré automatiquement teste toujours si la table qui va être définie existe déjà dans la base de donnée – créée au début du script.

Ensuite, le script précise le moteur de stockage utilisé pour chaque table (ENGINE).

Enfin, il précise également le nom des contraintes dans chaque table (CONSTRAINT).

Contrairement à mon script manuel, le script généré automatiquement par MySQL Workbench est plus précis, il tient en compte des spécificités de chaque table afin de réduire le nombre de variables générées automatiquement par MySQL.

3- Peuplement des tables

Voici le script de peuplement de la base de données :

```
CREATE TABLE tmp ( country VARCHAR, iso2 VARCHAR, iso3
VARCHAR, region_code INTEGER, region VARCHAR,
sub_region_code INTEGER, sub_region VARCHAR, disaster
VARCHAR, year INTEGER, number INTEGER) ;

\copy tmp(
country, iso2, iso3, region_code, region,sub_region_code,
sub_region, disaster, year, number)
FROM 'Climate_related_disasters_frequency.csv'
WITH CSV HEADER NULL '' ;
```

```
CREATE TABLE region (
region_code INTEGER PRIMARY KEY,
name VARCHAR) ; CREATE
sub_region_code
name TABLE sub_region (
region_code INTEGER PRIMARY KEY,
) ; CREATE VARCHAR,
country_code INTEGER REFERENCES region (region_code)
name
ISO2 TABLE country (
ISO3 SERIAL PRIMARY KEY,
sub_region_code VARCHAR,
) ; CREATE VARCHAR(2),
disaster_code VARCHAR(3),
disaster VARCHAR) ; CREATE TABLE climate_disaster (
INTEGER REFERENCES sub_region (sub_region_code)
country_code INTEGER REFERENCES country (country_code),
disaster_code INTEGER REFERENCES disaster (disaster_code),
year INTEGER NOT NULL,TABLE disaster (
PRIMARY KEY (country_code,disaster_code,year),SERIAL PRIMARY KEY,
number INTEGER) ;
```

```
INSERT INTO region
SELECT DISTINCT region_code, region
FROM tmp ;

INSERT INTO sub_region
SELECT DISTINCT sub_region_code, sub_region, region_code
FROM tmp ;

INSERT INTO country (name, ISO2, ISO3, sub_region_code)
SELECT DISTINCT country, iso2, iso3, sub_region_code
FROM tmp ; INSERT INTO disaster (disaster)
SELECT DISTINCT disaster
FROM tmp ; INSERT INTO climate_disaster (country_code, disaster_code,
year,
number)
SELECT country_code, disaster_code, year, number
FROM tmp
JOIN country ON name=country
JOIN disaster ON disaster.disaster=tmp.disaster ;
```

Pour commencer, nous définissons une table temporaire (tmp) qui va être remplie par les informations du fichier CSV. Elle va nous servir à réaliser des projections grâce à des requêtes.

Ensuite, nous définissons les tables region, sub_region, country, disaster et climate_disaster, conformément à l'exemple donné dans le sujet de cette SAE et en précisant le type SERIAL pour les clés primaires qui ne figurent pas dans le fichier CSV.

Une fois fait, nous allons commencer à remplir ces tables grâce à des requêtes et à la table temporaire accueillant le fichier CSV. L'ordre n'ayant que peu d'importance, nous allons les remplir dans l'ordre de leur création, en commençant par region et en finissant par `climate_disaster`.

Pour `region`, nous sélectionnons les colonnes region_code et region depuis tmp, en évitant la redondance grâce à DISTINCT.

Pour sub_region, nous faisons la même chose avec sub_region_code, sub_region et region_code, les colonnes remplissant respectivement les colonnes sub_region_code, `name et region_code de la table sub_region.`

Pour country, nous précisons les colonnes qui vont être remplies – soit name, ISO2, ISO3, sub_region_code – afin d'éviter qu'il n'y ait une erreur car le logiciel SQL essaie d'entrer une variable de type VARCHAR dans la clé primaire de type SERIAL.

Pour disaster, nous faisons la même chose que pour country mais avec la colonne `disaster.`

Enfin, pour climate_disaster, nous allons devoir réaliser des jointures afin d'intégrer country_code et disaster_code non définis dans la table temporaire. La jointure entre climate_disaster et country_code se fait en fonction du nom du pays (country.name) et celle entre climate_disaster et disaster se fait en fonction du nom de la catastrophe
`(disaster.disaster)`.

Dans cette SAE nous avons vu les différences entre les associations fonctionnelles et maillées du cours et de l'AGL MySQL Workbench, ainsi que la manière de créer et de peupler des tables d'une base de données à partir d'un fichier CSV.