Rezzag-Mahcene Youssouf
Boreas

Rapport SAE « Création d'une base de données »

Sommaire :

1.Script Manuel de création de la base de données

Création de la table Région.

```
CREATE TABLE Region (
   region_code INT PRIMARY KEY,
   name VARCHAR(255) NOT NULL
);
```

Création de la table status.

```
CREATE TABLE Status (
  status VARCHAR(255) PRIMARY KEY
);
```

Création de la table Country

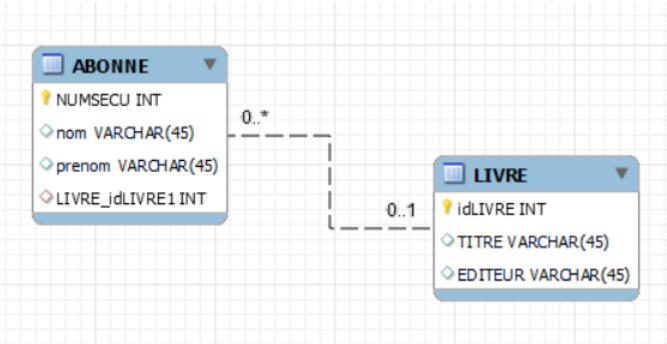
```
CREATE TABLE Country (
   id_country INT PRIMARY KEY,
   name VARCHAR(255) NOT NULL,
   region_code INT,
   is_ldc BOOLEAN,
   FOREIGN KEY (region_code) REFERENCES
Region(region_code)
);
```

Création de la table freedom

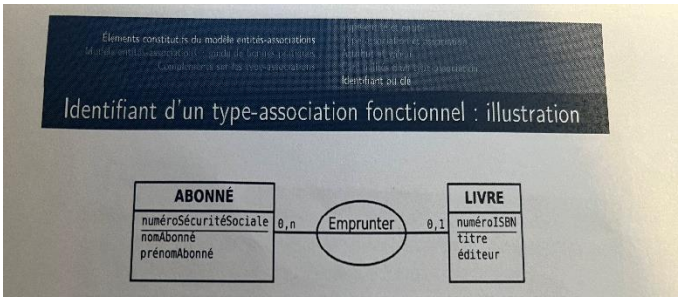
```
CREATE TABLE Freedom (
   id_country INT,
   year INT,
   civil_liberties INT,
   political_rights INT,
   status VARCHAR(255),
   PRIMARY KEY (id_country, year),
   FOREIGN KEY (id_country) REFERENCES
Country(id_country),
   FOREIGN KEY (status) REFERENCES
Status(status)
);
```

2.Modélisation et script de création « avec AGL »

2.1 Illustrations comparatives cours/AGL commentée d'une association fonctionnelle.



Exemple d'une association fonctionnelle réalisé avec l'AGL - (MySQL a été utilisé pour tout le projet)
Figure n°1
(On réalise le même schéma pour plus de clarté)



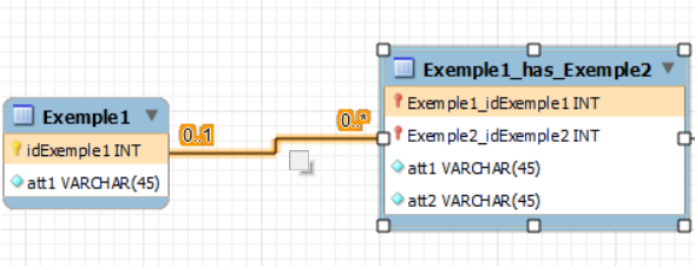
Exemple du cours d'une association fonctionnelle.
Figure n°2

Dans cette comparaison, nous examinerons les différences entre le modèle de l'AGL et un modèle traditionnel de cours, en nous concentrant sur les attributs, les cardinalités et les associations. Cela nous aidera à mieux comprendre les forces et les particularités de chaque approche en modélisation des données.

1. Les attributs :

Avec l'AGL, il est possible de visualiser chaque attribut en détaillant son nom, son type (INT, VARCHAR, etc.) et d'utiliser des symboles distincts. Par exemple, une clé jaune (NUMSECU) indique une clé primaire, tandis qu'un point rouge signifie une clé étrangère.

La clé étrangère est également représentée comme un attribut dans la modélisation. Dans cet exemple, la clé étrangère est liée à l'entité `ABONNE' Cette liaison permet d'établir une connexion entre les deux entités, offrant une approche pratique pour modéliser des relations spécifiques. Lorsque l'on survole la relation avec la souris, la clé étrangère s'illumine.



Dans un Modèle Conceptuel de Données (MCD), seuls les libellés des attributs sont visibles, les clés étrangères ne sont pas explicitement illustrées. La clé primaire est généralement mise en évidence par un soulignement. Les associations et les cardinalités sont définies, par exemple, une cardinalité "0,1" indique qu'une entité peut avoir zéro ou au maximum une relation tandis que "0, n" indique zéro ou plusieurs relations.

2. Les associations et cardinalités :

AGL :

- L'emplacement des cardinalités sont inversées par rapport au MCD

- La cardinalité « 0,1 » est positionné à côté de EXEMPLE1 et « 0,n » à côté de EXEMPLE1_HAS_EXEMPLE2

- Dans une association fonctionnelle, les emplacements où les cardinalités sont indiquées sont inversés par rapport au MCD.

- Pour le MCD, « 0,1 » est positionné à côté de LIVRE ; « 0, n » est positionné à côté d'ABONNE.

- En AGL, « 0,1 » est positionné à côté d'ABONNE ; « 0, n » est positionné à côté de LIVRE.

- Cependant, le concept reste le même que dans le MCD concernant le nombre possible de relations entre les classes : dans la classe LIVRE, il peut y avoir soit aucune relation, soit au maximum une relation ; dans la classe ABONNE, il peut y avoir soit aucune relation, soit plusieurs relations une relation, class ABONNE peut avoir sois zéro relation ou bien plusieurs relations.

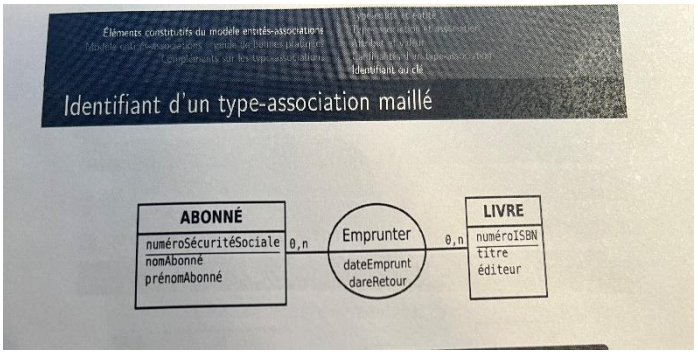
MCD :

- Les deux entités sont reliées par une ligne droite et un type association.

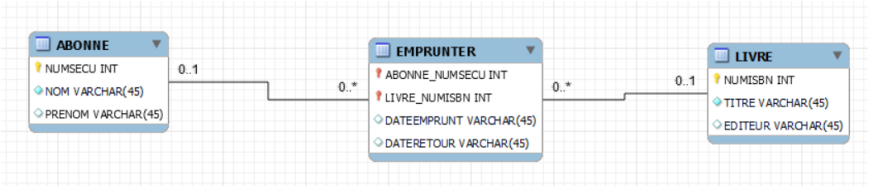
- Deux cardinalités '0.n' et '0.1'

On sait que la cardinalité "0,1" indique qu'une entité peut ne pas avoir de relation du tout avec une autre entité, ou au maximum, elle peut avoir une seule relation. Et que la cardinalité "0,n" indique qu'une entité peut ne pas avoir de relation du tout avec une autre entité, ou elle peut avoir plusieurs relations.

2.2 Illustrations comparatives cours/AGL commentée d'une association maillée.



Exemple d'une association maillée (MCD)
Figure n°1



Exemple d'une association maillée réalisé avec l'AGL
Figure n°2

Concernant les attributs, on remarque les mêmes différences que pour l'association fonctionnelle.

Pour les associations et cardinalités :

- Les cardinalités « 0,n » sont affichés des deux cotés de la relation, cela signifie que les deux entités peuvent avoir soit aucune soit une infinité de relations entre elles.
- Il y a quatre cardinalités dans le schéma de l'AGL et deux pour le MCD,

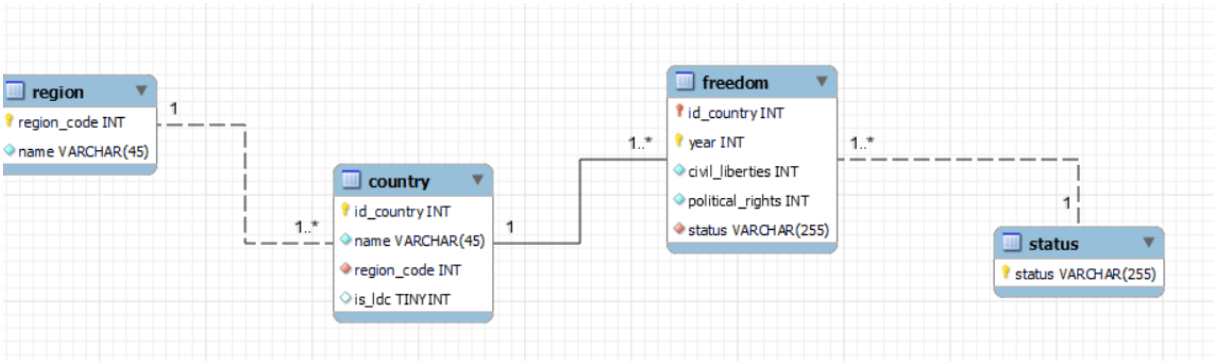
Dans l'AGL, pour l'association mailée, une nouvelle classe est créée automatiquement, les clés primaires des deux entités deviennent clés primaires et étrangères de cette nouvelle classe.

Remarque :

On peut clairement établir ces égalités entre le MCD et l'AGL :

1 = 1.1
1.n = 1.*
0.n = 0.*
0.1 = 0.1

2.3 Modèle physique de données réalisé avec l'AGL.



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

```
1  CREATE TABLE `country` (
2    `idcountry` int NOT NULL,
3    `name` varchar(45) DEFAULT NULL,
4    `region_code` int DEFAULT NULL,
5    `is_ldc` tinyint DEFAULT NULL,
6    PRIMARY KEY (`idcountry`),
7    KEY `region_code_idx` (`region_code`),
8    CONSTRAINT `region_code` FOREIGN KEY (`region_code`) REFERENCES `region` (`region_code`)
9  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

CREATE TABLE 'country' (
'idcountry' int NOT NULL,
'name' varchar(45) DEFAULT NULL,
'region_code' int DEFAULT NULL,
'is_Idc' tinyint DEFAULT NULL, PRIMARY KEY ('idcountry'),
KEY 'region_code_idx' ('region_code'),
CONSTRAINT 'region _code' FOREIGN KEY ('region_code') REFERENCES 'region' ('
region_code')
) ENGINE= InnODB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```
CREATE TABLE `freedom` (
  `id_country` int NOT NULL,
  `year` int NOT NULL,
  `civil_liberties` int DEFAULT NULL,
  `political_rights` int DEFAULT NULL,
  `status` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_country`,`year`),
  KEY `status_idx` (`status`),
  CONSTRAINT `id_country` FOREIGN KEY (`id_country`) REFERENCES `country` (`idcountry`),
  CONSTRAINT `status` FOREIGN KEY (`status`) REFERENCES `status` (`status`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

CREATE TABLE 'freedom' (
‘id_country’ int NOT NULL,
‘year’ int NOT NULL,
‘civil _liberties’ int DEFAULT NULL,
‘political _rights’ int DEFAULT NULL,
'status’ varchar(45) DEFAULT NULL,
PRIMARY KEY (‘id _country’, ‘year’),
KEY ‘status_idx’ ('status'),
CONSTRAINT ‘id_country’ FOREIGN KEY ('id_country’) REFERENCES ‘country’ ('idcountry’),
CONSTRAINT ‘status’ FOREIGN KEY ('status’) REFERENCES 'status’ ('status’)
) ENGINE-InnoDB DEFAULT CHARSET=utf8mb4 COLLATE-utf8mb4_0900_ai_ci

```
1  CREATE TABLE `status` (
2    `status` varchar(255) NOT NULL,
3    PRIMARY KEY (`status`)
4  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

CREATE TABLE ' status’ (
'status’ varchar(255) NOT NULL, PRIMARY KEY ('status’)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```
1  CREATE TABLE `region` (
2    `region_code` int NOT NULL,
3    `name` varchar(45) DEFAULT NULL,
4    PRIMARY KEY (`region_code`)
5  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

CREATE TABLE 'region’ (

‘region_code’ int NOT NULL,

‘name’ varchar(45) DEFAULT NULL,

PRIMARY KEY (‘region_code’)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

2.5 Discussions sur les différences entre les scripts produits manuellement et automatiquement

- On remarque que chaque nom d'attributs ou de tables est entre ' '.

- L'ajout de '**DEFAULT NULL'** cela signifie que tant qu'aucune valeur n'est entrée la valeur est NULL

- Malgré la présence d'une seule clé primaire, elle est tout de même mentionné en fin de requête entre parenthèse.

- L'ajout de la clause **CONSTRAINT**, elle est utilisée pour déclarer des règles qui garantissent l'intégrité des données dans une table en imposant des conditions spécifiques sur les valeurs des colonnes.

- La clause **ENGINE** spécifie le moteur de stockage à utiliser pour la table. Chaque système de gestion de base de données peut avoir différents moteurs de stockage qui ont des caractéristiques spécifiques. Par exemple, MySQL propose des moteurs de stockage tels que InnoDB.

- La clause **DEFAULT CHARSET** spécifie le jeu de caractères par défaut pour la table. Le jeu de caractères détermine le jeu de caractères utilisé pour stocker les données textuelles dans la table. Par exemple, UTF-8 est un jeu de caractères couramment utilisé qui prend en charge une large gamme de caractères internationaux.

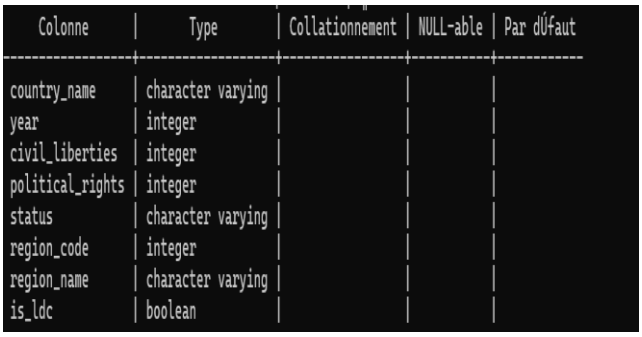
- La clause **COLLATE** spécifie le classement à utiliser pour trier et comparer les données de la table. Le classement définit l'ordre dans lequel les caractères sont triés, ce qui peut avoir un impact sur les opérations de tri et de comparaison.

3.Peuplement des tables

3.1 Description commentée des différentes étapes de votre script de peuplement.

Pour cette partie, nous allons utiliser SQL SHELL :

Premièrement au démarrage du SHELL, on commence par créer une table que l'on nomme « temporaire », avec tous les attributs dans laquelle, nous allons stocker toutes les données du fichier csv.



Colonne	Type	Collationnement	NULL-able	Par dÚfaut
country_name	character varying			
year	integer			
civil_liberties	integer			
political_rights	integer			
status	character varying			
region_code	integer			
region_name	character varying			
is_ldc	boolean			

```
CREATE TABLE temp (
        country_name VARCHAR,
        year INT,
        civil_liberties INT,
        political_rights INT,
        status VARCHAR,
        region_code INTEGER,
        region_name VARCHAR,
        is_ldc BOOLEAN
);
```

On effectue le transfert vers la table temporaire en utilisant :

```
\copy temporaire FROM 'C:\Users\Youssouf\Downloads\freedom\freedom.csv' WITH CSV HEADER DELIMITER ',' ;
```

Si le message : « COPY 4979 » s'affiche, cela signifie que le transfert c'est bien déroulé.

Pour le peuplement des tables, on doit suivre un ordre bien précis.

- Création de la table region
- Création de la table country
- Création de la table status
- Création de la table freedom

Nous allons utiliser la requête suivante pour la table region:

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

```
postgres=# INSERT INTO region(region_code,name) SELECT DISTINCT region_code, region_name FROM temporaire;
INSERT 0 5
postgres=# select * from region;
 region_code |   name
-------------+----------
         142 | Asia
           9 | Oceania
          19 | Americas
         150 | Europe
           2 | Africa
(5 lignes)
```

Nous allons utiliser la requête suivante pour la table country :

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

```
postgres=# INSERT INTO country(name,region_code,is_ldc) SELECT DISTINCT country,region_code,is_ldc FROM temporaire;
INSERT 0 193
postgres=# SELECT * FROM country;
 id_country |                        name                        | region_code | is_ldc
------------+----------------------------------------------------+-------------+--------
       4980 | Monaco                                             |         150 |      0
       4981 | Croatia                                            |         150 |      0
       4982 | Nauru                                              |           9 |      0
       4983 | Zambia                                             |           2 |      1
       4984 | Montenegro                                         |         150 |      0
       4985 | Indonesia                                          |         142 |      0
       4986 | Yemen                                              |         142 |      1
```

Nous allons utiliser la requête suivante pour la table status :

```
INSERT INTO status(status) SELECT DISTINCT status FROM temporaire ;
```

```
postgres=# INSERT INTO status(status) SELECT DISTINCT status FROM temporaire;
INSERT 0 3
postgres=# select * from status;
 status
--------
 PF
 NF
 F
(3 lignes)
```

Et pour finir nous allons utiliser la requête suivante pour la table freedom :

```
INSERT INTO freedom (id_country, year, civil_lib+erties, politcal_rights,status) SELECT country.id_country,year,civil_liberties,politcal_rights,status FROM temporaire JOIN country ON country.name = temporaire.country ;
```

```
postgres=# INSERT INTO freedom(id_country,year,civil_liberties,political_rights,status) SELECT country.id_country,year,cl,pr,status FROM temporaire JOIN cou
ntry ON country.name = temporaire.country;
INSERT 0 4979
postgres=# SELECT * FROM freedom;
 id_country | year | civil_liberties | political_rights | status
------------+------+-----------------+------------------+--------
       5813 | 1995 |               7 |                7 | NF
       5813 | 1996 |               7 |                7 | NF
       5813 | 1997 |               7 |                7 | NF
       5813 | 1998 |               7 |                7 | NF
       5813 | 1999 |               7 |                7 | NF
       5813 | 2000 |               7 |                7 | NF
       5813 | 2001 |               7 |                7 | NF
```

La petite particularité pour cette table est d'utiliser JOIN car id_country n'est pas dans la table temporaire.