Ben Mansour

Habib

Boréas

SAE :

Base de Données : Freedom in the world

I. Sommaire

II. Contexte de la SAE

Contexte générale (Modélisation de la base de données)

III. Description du travaille demandé

A. Script Manuelle

Script SQL de création de tables

B. Modélisation et script de création < avec AGL>

Illustration comparatives cours/AGL commenté d'une association fonctionnelle

Illustration comparatives cours/AGL commenté d'une association maillée

Modèle physique de données réalisé avec AGL

Discussion sur les différences entre les scripts produits manuellement et automatiquement

Peuplement des tables

IV. Modalités, ressources et barèmes

Format du rapport

Echéances

Barème

Travaillé demandé :

1) INSERT INTO country(nom,is_ldc,region_code)SELECT DISTINCT country,is_ldc,region_code FROM tmp;

```
Région CREATE TABLE  (     code_région INTEGER PRIMARY KEY,     nom
VARCHAR(255) NOT NULL) ;



CREATE  TABLE  country  (
    id_country SERIAL PRIMARY KEY,
    nom VARCHAR(45) NOT NULL,
    region_code INTEGER REFERENCES region(region_code) ON DELETE SET NULL,
    is_ldc INTEGER NOT NULL CHECK(is_ldc >=0 AND is_ldc<=1) ) ;

CREATE TABLE  Statut    (statut VARCHAR(2) PRIMARY KEY CHECK (statut
IN('NF','PF','F')) ) ;

CREATE  TABLE  Freedom (
    années INTEGER,
    id_country INTEGER REFERENCES country (id_country) ON DELETE CASCADE,
    civil_liberties INTEGER NOT NULL CHECK(civil_liberties >=1 AND
civil_liberties <=7),
    Political_rights INTEGER NOT NULL CHECK( Political_rights >=1 AND
Political_rights <=7 ),
    statut VARCHAR(2) REFERENCES statut (statut) ON DELETE SET NULL,
    PRIMARY KEY (years,id_country) ) ;
```

Explication ☺ (je tiens a précise Monsieur que lorsque j'ai explique une commande je ne trouve pas cela important de le répété)

Tout d'abord, afin de réaliser le script de la modélisation, j'ai créé une table pour chaque type d'entité. Ensuite, pour chacune, j'ai introduit leurs attributs avec leur type.

Table région :

Ici pour l'attribut region_code de la table région, étant donné que c'est un code, j'ai dû lui intégrer la commande INTEGER, et pour le nom VARCHAR NOT NULL, car c'est une chaîne de caractères qui ne doit pas être nulle, c'est-à-dire avoir une valeur obligatoire. De plus, il faut préciser que la clé primaire ici est region_code avec la commande PRIMARY KEY. On sait cela par le soulignement de cet attribut.

Table "country" :

Pour le cas de la table "country", pour id country, je n'ai pas seulement dit que c'était une clé, mais aussi qu'elle était de type SERIAL, c'est-à-dire auto-incrémenté, afin de garantir l'unicité des identifiants de chaque pays étant donné qu'on avait une cardinalité de 1 à 1.

De plus, dans cette table, nous avons utilisé une nouvelle commande INTEGER REFERENCES pour préciser que region_code est une clé étrangère provenant de la table "région". Cela signifie que chaque valeur de

region_code dans la table "country" doit correspondre à une valeur existante dans la table "région". Cela empêche la création de relations incohérentes entre les régions et les pays. Et la contrainte ON DELETE SET NULL indique que si la région associée à un pays est supprimée, la référence à cette région dans le pays sera définie sur NULL. Enfin, pour l'attribut is_ldc, j'ai utilisé CHECK(contrainte) pour dire que l'attribut doit avoir soit la valeur 0 soit 1 (information tableau).

Table "statuts" :

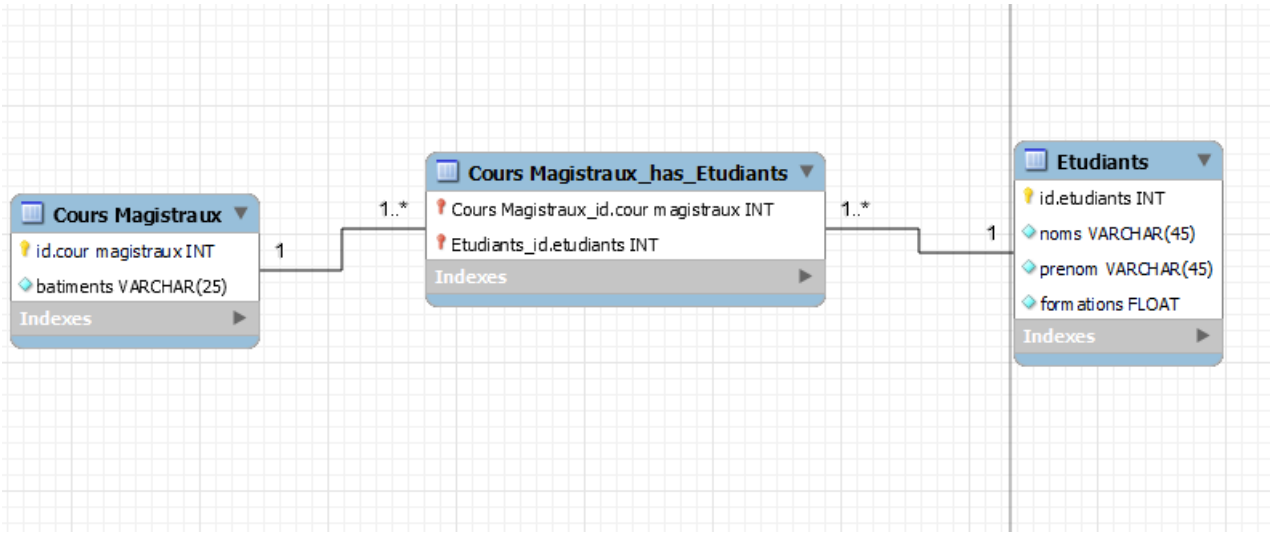
Pour la table "statuts", j'ai simplement indiqué que l'attribut statuts devait avoir comme chaîne de caractères uniquement celles-ci ('NF', 'PF' ou 'F').

Table "freedom" :

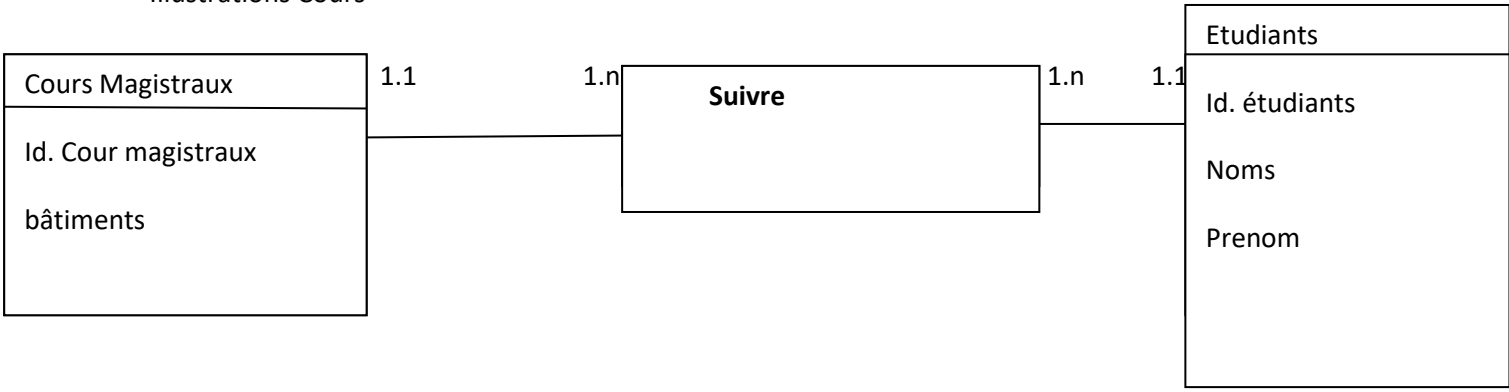
Pour finir, la table "freedom" a réutilisé les commandes précédentes principalement avec CHECK pour définir les limites, avec civil_liberties et political rights qui devaient être compris entre 1 et 7 (informations tableau). Néanmoins, il y a une nouveauté dans cette table pour indiquer une contrainte ON DELETE CASCADE.

2) Association fonctionnelle :

Illustrations AGL



Illustrations Cours



Explication de l'exemple :

Nous avons simplement des étudiants partant suivre/assister a un cours. Et un cours suivit par plusieurs étudiants.

Commentaire :
On peut constater plusieurs différences entre l'illustration manuelle et celle de l'AGL. On peut voire que l'AGL pour certains points est plus précis notamment aux niveaux de l'indication des types des attributs. On constate qu'il indique après chaque attribut s'il s'agit d'une chaine avec son nombre d'éléments ou d'une liste de nombre. De plus on peut voir qu'il met des figures devant chaque attribut pour indiquer si c'est clé primaire ou étrangers ou un simple attribut. On peut aussi voire que sur l'AGL que la cardinalité est affiche différemment au lieu de n il ya des étoiles et lorsque que la cardinalité est de 1.1 il affiche une seule fois. Enfin on constate que pour les types associations il ya l'affichage de clés étrangères.

3) <u>Association maillé</u>:

Illustrations AGL

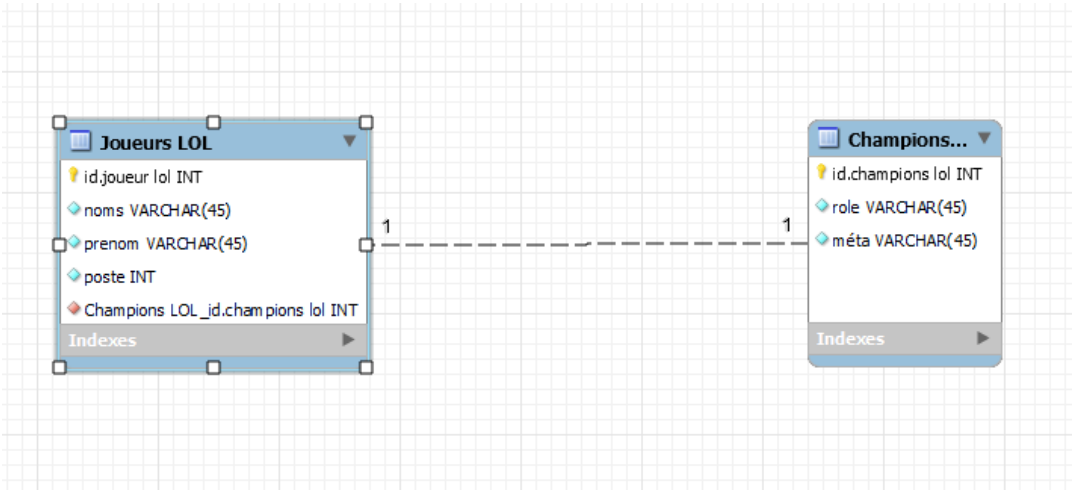
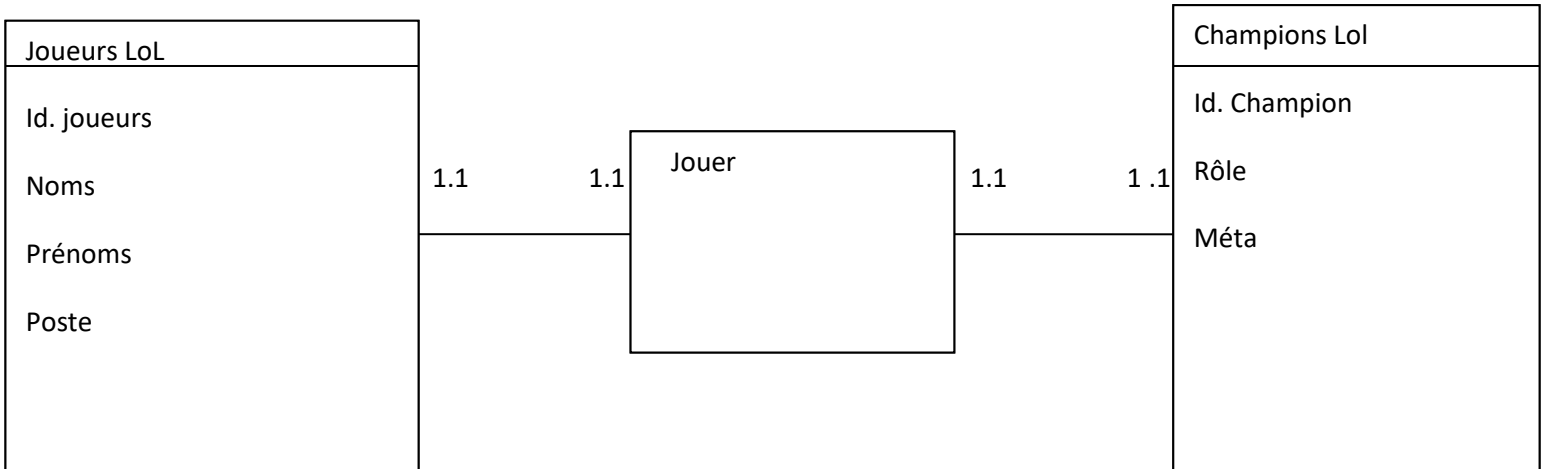


Illustration cours



Explication de l'exemple :

Nous avons simplement un joueurs qui joue un champions de LoL sur une partie de jeu (Game vu que vous connaissez monsieur ☺) .Et un champion qui peut être jouer que par un seul joueur d'une équipe.(+1 pour l'originalité si vous êtes simpatiques)

Commentaire :
Comme pour l'ancien commentaire à l'exception de certaines choses .On constate plusieurs différences. Ici pour l'illustrations de l'AGL le type association n'est pas représenté ainsi que les traits sont en pointillé et non en traits plain. Ce la veut simplement dire que c'est une relation identifiant les plus classique avec dans l'exemple comme cardinalité maximale de 1.Pour savoir une association non identifiant est une relation entre deux entités où l'entité dépendante n'a pas de clé primaire propre. Cela signifie que l'entité dépendante est identifiée uniquement par sa relation avec l'entité principale à laquelle elle est associée.

4)

```
CRÉER UNE TABLE SI NON EXISTE `question3`.`region` (
  `id.region code` INT NOT NULL,
  `nom` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id.region code`))
MOTEUR = InnoDB ;
CRÉER UNE TABLE SI N'EXISTE PAS ` question3`.`Country` (
  `idCountry` INT NOT NULL,
  `is_ldc` INT NULL,
  `region_id.region code` INT NULL,
  `region_id.region code1` INT NULL,
  PRIMARY KEY (`idCountry` ),
  CONTRAINTE `fk_Country_region1`
    CLÉ ÉTRANGÈRE (`region_id.region code1`)
    RÉFÉRENCES `question3`.`region` (`id.region code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
CRÉER UN INDEX `fk_Country_region_idx` ON `question3`.`Country`
(`region_id.region code` ASC) VISIBLE ;
CRÉER UN INDEX `fk_Country_region1_idx` SUR `question3`.`Country`
(`region_id.region code1` ASC) VISIBLE ;
CRÉER UNE TABLE SI N'EXISTE PAS ` question3`.`Status` (
  `idStatus` INT NOT NULL,
  PRIMARY KEY (`idStatus`))
MOTEUR = InnoDB ;
CRÉER UNE TABLE SI NON EXISTE `question3`.`Liberté` (
  `année` INT NON NULL,
  `civil_liberties` INT NON NULL,
  `politique_droits` INT NULL,
```

```
  `Freedomcol` VARCHAR(45) NULL,
  `Country_idCountry` INT NOT NULL,
  ` Status_idStatus` INT NON NULL,
  `Country_idCountry1` INT NON NULL,
  `Status_idStatus1` INT NON NULL,
  CLÉ PRIMAIRE (`année`, `Country_idCountry`, `Country_idCountry1`),
  CONTRAINTE `fk_Freedom_Country2`
    CLÉ ÉTRANGÈRE (`Country_idCountry1`)
    RÉFÉRENCES `question3 `.`Country` (`idCountry`)
    ON SUPPRIMER AUCUNE ACTION
    SUR LA MISE À JOUR AUCUNE ACTION,
  CONTRAINTE `fk_Freedom_Status2`
    CLÉ ÉTRANGÈRE (`Status_idStatus1`)
    RÉFÉRENCES `question3`.`Status` (`idStatus`)
    ON SUPPRIMER AUCUNE ACTION
    SUR LA MISE À JOUR NON ACTION)
MOTEUR = InnoDB ;

CRÉER UN INDEX `fk_Freedom_Country1_idx` SUR `question3`.`Freedom`
(`Country_idCountry` ASC) VISIBLE ;

CRÉER UN INDEX `fk_Freedom_Status1_idx` SUR `question3`.`Freedom`
(`Status_idStatus` ASC) VISIBLE ;

CRÉER UN INDEX `fk_Freedom_Country2_idx` SUR `question3`.`Freedom`
(`Country_idCountry1` ASC) VISIBLE ;

CRÉER UN INDEX `fk_Freedom_Status2_idx` SUR `question3`.`Freedom`
(`Status_idStatus1` ASC) VISIBLE ;

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

Liste des différences

1) Les noms des tables et des colonnes diffèrent entre les deux scripts.

2) Le premier script utilise des contraintes CHECK dans les déclarations de colonnes pour les valeurs spécifiques, tandis que le deuxième script utilise des contraintes CHECK dans les déclarations de table.

3) Le deuxième script utilise le moteur InnoDB pour toutes les tables, tandis que le premier script n'a pas spécifié de moteur (ce qui signifie généralement que le moteur par défaut sera utilisé).

4) Les noms des clés étrangères et des index diffèrent entre les deux scripts.

5) Méthode du peuplement des tables :

Etapes :

- Création d'une table temporaire appelé 'tmp' contenant tous les attributs du fichier plat csv.

```
CREATE TABLE tmp (
    pays VARCHAR(255),
    années INTEGER,
    libertés_civiles INTEGER,
    droits_politiques INTEGER,
    statut VARCHAR(255),
    code_région INTEGER,
    nom_région VARCHAR(255),
    is_ldc INTEGER
) ;
```

- J'ai mis dans la table temporaire les valeurs associés aux colonnes du fichier plat pour qu'après je doive simplement remplir les tables une par une grâce a la commande :

COPY tmp(country,years,civil_liberties,political_rights,status,region_code,nom,is_ldc)FROM lien du fichier plats DELIMITER,CSV HEADER

- Puis j'ai rempli les tables dans un ordre précis afin de ne pas avoir de clés étrangères qui ne sont référencé dans aucune table.
  Commande :

INSERT INTO nom_table(A)(colA,colA)SELECT DISTINCT colB,colB FROM nom_table(B)

- La commande citée au- dessus nous a permis de peupler les tables. On a eu besoin de l'instruction INSERT INTO permettant de peupler la table a partir des données de la table temporaire contenant les données du fichier CSV. Enfin on utilise SELECT DISTINCT pour n'avoir que les informations qu'on souhaite et en plus non doublé.
- ET si on souhaite être sur que sa marche on affiche la table avec cette commande

SELECT*FROM nom_table ;

- Désormais que nous avons cette structure nous allons pouvoir peupler nos tables une par une.

(a) Table region :

INSERT INTO region(region_code,nom)SELECT DISTINCT region_code,region_name FROM tmp ;

Vérification : SELECT*FROM region ;

	region_code [PK] integer	nom character varying (45)
1	142	Asia
2	9	Oceania
3	19	Americas
4	150	Europe
5	2	Africa

(b) Table country :

INSERT INTO country(nom,is_ldc,region_code)SELECT DISTINCT country,is_ldc,region_code FROM tmp ;

Vérification : SELECT*FROM country ;

	id_country [PK] integer	nom character varying (255)	region_code integer	is_ldc integer
1	147	Libya	2	0
2	148	Bahamas	19	0
3	149	Bosnia and Herzegovina	150	0
4	150	France	150	0
5	151	Panama	19	0
6	152	Gabon	2	0
7	153	Denmark	150	0
8	154	Peru	19	0
9	155	Maldives	142	0
10	156	Eswatini	2	0
11	157	Mauritius	2	0
12	158	Jordan	142	0
13	159	Norway	150	0
14	160	Iraq	142	0
15	161	Montenegro	150	0
16	162	Uganda	2	1

X

(c) Table status :

INSERT INTO status(statuts)SELECT DINSTINCT status FROM tmp ;

Vérification : SELECT*FROM status ;

	statuts [PK] character varying (2)
1	PF
2	NF
3	F

(d) Table freedom :

- Ici nous avons un cas un petit peu spéciale du fais que les clés de country ne sont comprises dans le fichier plats alors il faut réussir a correspondre les bonnes lignes de donnés aux id country. Plus précisément que les informations du pays soit bien associés à l'identifiants du pays correspondants. Pour cela nous allons utiliser une jointure entre tmp et country et avec country =nom

La commande :

INSERT INTO freedom(id_country,year,civil_liberties,political_rights,status)SELECT country.id_country,tmp.year,tmp.civil_liberties,tmp.political_rights,tmp.status FROM tmp,country WHERE country=name ;

Vérification :SELECT*FROM freedom ;

(Il ne faut pas oublier de supprimer la table tmp avec DROP TABLE étant donné qu'elle ne fais pas partie du schéma de base)

	years [PK] integer	id_country [PK] integer	civil_liberties integer	political_rights integer	statut character varying (2)
1	1995	531	7	7	NF
2	1995	338	7	7	NF
3	1996	531	7	7	NF
4	1996	338	7	7	NF
5	1997	531	7	7	NF
6	1997	338	7	7	NF