Ben Mansour

Habib

Boréas

SAE :

Base de Données : Freedom in the world

Sommaire

# Contexte de la SAE

Contexte générale (Modélisation de la base de données)

# Description du travaille demandé

## Script Manuelle

Script SQL de création de tables

## 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

# 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" :

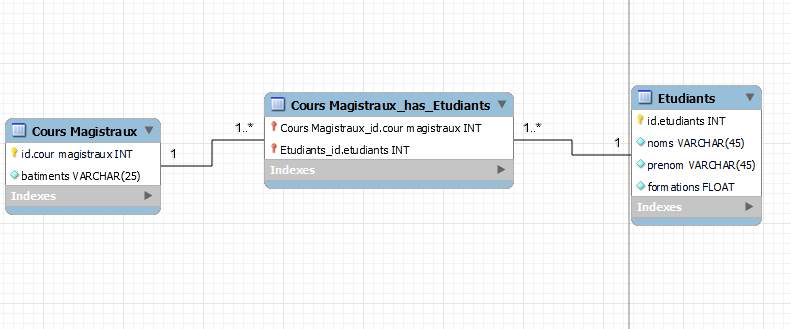
Pour la table "statuts", j'ai simplement indiqué que l'attribut statuts devait avoir comme chaîne de caractères uniquement celles-ci ('NF', 'PF' ou 'F').

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

Etudiants

Id. étudiants

Noms

Prenom

Cours Magistraux

Id. Cour magistraux

bâtiments

1.n 1.1 1.n 1.n 1.1 1.n 1.1

**Suivre**

Explication de l’exemple :

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

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.

Commentaire :

3) Association maillé:

Illustrations AGL

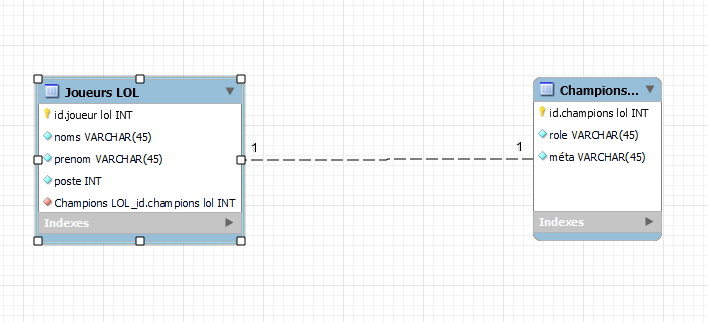


Illustration cours

Joueurs LoL

Id. joueurs

Noms

Prénoms

Poste

Champions Lol

Id. Champion

Rôle

Méta

Jouer

1.1 1.1 1.1 1 .1

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. 1) Les noms des tables et des colonnes diffèrent entre les deux scripts. 2. 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. 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. 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

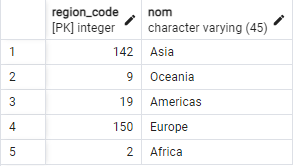
SELECT\*FROM nom\_table ;

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

###### Table region :

INSERT INTO region(region\_code,nom)SELECT DISTINCT region\_code,region\_name FROM tmp ;

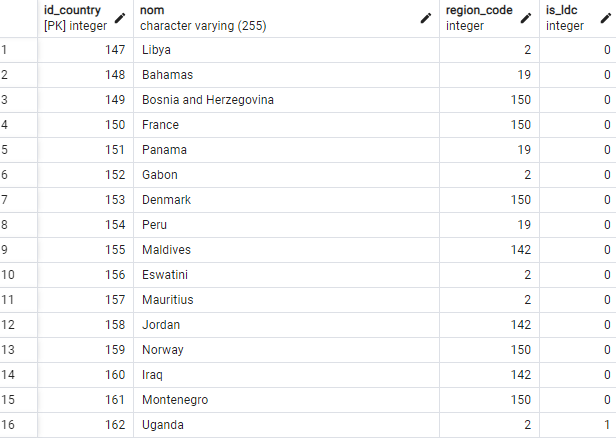
Vérification : SELECT\*FROM region ;



###### Table country :

INSERT INTO country(nom,is\_ldc,region\_code)SELECT DISTINCT country,is\_ldc,region\_code FROM tmp ;

Vérification : SELECT\*FROM country ;

x

###### Table status :

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

Vérification : SELECT\*FROM status ;

###### 

###### 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)

