Bases de Données et langage SQL
Situation d'Apprentissage et d'Évaluation
Conception et implémentation d'une base de données

SAÉ 2-01

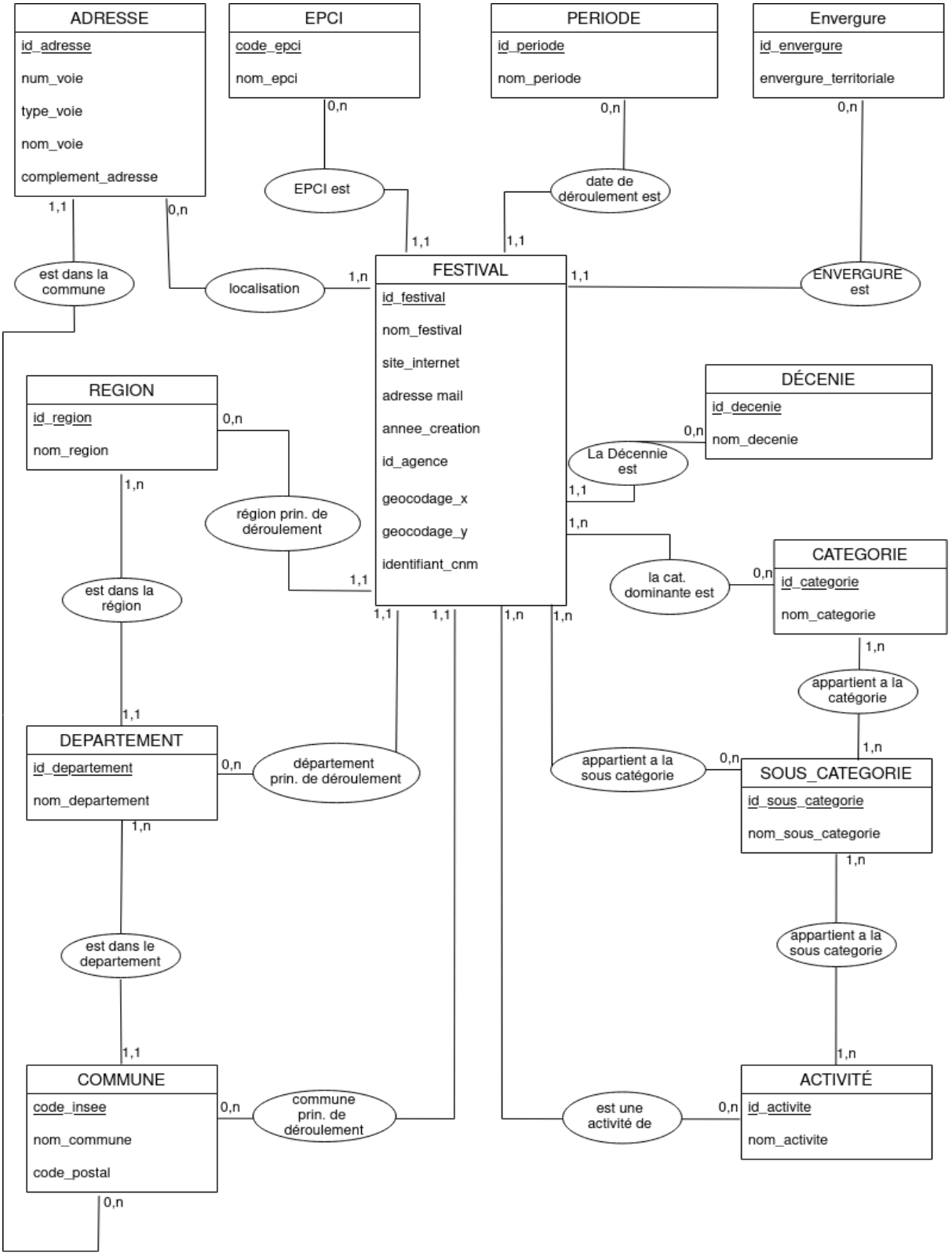
29/05/2024
BUT1 SD
Groupe : Galton

Avenel Kyllian
Diallo Thierno

SOMMAIRE

I. ETAPE 1 : Modélisation entités-associations et relationnelle

1. Modèle entités-associations



2. Schéma relationnel

EPCI (code_epci, nom_epci)

ENVERGURE (id_envergure, envergure_territoriale)

REGION (id_region, nom_region)

DEPARTEMENT (id_departement, nom_departement, id_region)

COMMUNE (code_insee, nom_commune,code_postal, id_departement)

ADRESSE (id_adresse, num_voie, nom_voie, type_voie, complement_adresse, code_insee)

CATEGORIE (id_categorie, nom_categorie)

SOUS_CATEGORIE (id_sous_categorie, nom_sous_categorie)

ACTIVITE (id_activité, nom_activité)

DECENIE (id_decenie, nom_decenie)

DATE_DEROULEMENT (id_periode, nom_periode)

FESTIVAL(id_festival, nom_festival, site_internet,
adresse_mail, annee_creation,
identifiant_agence, identifiant_cnm,
geocodage_x, geocodage_y,
id_envergure, code_epci, id_periode
id_decenie, code_insee)

ADRESSE_FESTIVAL (id_adresse, id_festival)

CATEGORIE_FESTIVAL (id_categorie, id_festival)

SOUS_CATEGORIE_FESTIVAL (id_sous_categorie, id_festival)

ACTIVITE_FESTIVAL (id_activite, id_festival)

SOUS_CATEGORIE_ACTIVITE (id_sous_categorie, id_activite)

CATEGORIE_SOUS_CATEGORIE (id_categorie, id_sous_categorie)

3. Description des entités, associations, tables (les attributs, les clés)

- Entité CATEGORIE : l'Entité categorie contient les différentes catégories possibles pour un festival, elle est composée de deux attributs à savoir :
    - id_categorie (INTEGER) : c'est un identifiant pour l'entité categorie, il est unique et non null (PRIMARY KEY)
    - nom_categorie (VARCHAR) : c'est le nom de la catégorie

- Entité SOUS_CATEGORIE : l'Entité sous catégorie contient les différentes sous catégories possibles pour un festival, elle est composée de trois attributs à savoir :
    - id_sous_categorie (INTEGER) : c'est un identifiant pour la sous catégorie, il est unique(PRIMARY KEY)
    - nom_sous_categorie (VARCHAR) : c'est le nom de la sous catégorie
    - id_categorie (INTEGER) : c'est une clé étrangère qui fait référence à id_categorie dans l'Entité categorie. il permet d'identifier la catégorie principale du festival à la quel la sous catégorie est liée

- Entité ACTIVITE : l'Entité activite contient les différentes activités possibles pour un festival, elle est composée de deux attributs à savoir :
    - id_activite (INTEGER) : c'est un identifiant pour l'Entité activite, il est unique (PRIMARY KEY)
    - nom_activite (VARCHAR) : c'est le nom de l'activité

- Entité EPCI : l'entité EPCI contient les établissements publics de coopération intercommunale et est composée de deux attributs :
    - code_epci (INTEGER) : le code epci est aussi l'identifiant de l'Entité EPCI (PRIMARY KEY)
    - nom_epci (VARCHAR) : cet attribut est une chaîne de caractères des noms des différents epci

- Entité ENVERGURE : l'Entité d'envergure contient les différents types de territoire ou les festivals sont représentés . Cette Entité compte deux attributs :
    - id_envergure (INTEGER) : c'est l'identifiant de l'Entité envergure et est unique (PRIMARY KEY)
    - envergure_territoriale (VARCHAR) : représente le type de territoire sur laquelle les festivals sont réalisés

- Entité ADRESSE : l'Entité adresse contient les informations géographiques précises où les festivals sont réalisés. Cette Entité compte six attributs :
    - id_adresse (INTEGER) : c'est l'identifiant de l'Entité, il est unique (PRIMARY KEY)
    - num_voie (VARCHAR) : cet attribut est le numéro de voie de la zone géographique du festival
    - type_voie (VARCHAR) : cet attribut est le type de voie où se situe le festival (rue, avenue, boulevard)
    - nom_voie (VARCHAR) : cet attribut est le nom de voie où se situe le festival

    - complement_adresse (VARCHAR) : chaîne de caractères du complément de l'adresse
    - code_insee (INTEGER) : c'est une clé étrangère qui fait référence à code_insee dans l'entité COMMUNE

- Entité REGION : l'Entité région contient les informations sur la région du déroulement des festivals. Cette Entité compte deux attributs :
    - id_region (INTEGER) : identifiant de l'Entité région, est aussi unique (PRIMARY KEY)
    - nom_region (VARCHAR) : chaîne de caractères contenant le nom des différentes régions.

- Entité DEPARTEMENT : l'Entité département contient les informations sur le département du déroulement des festivals. Cette Entité compte trois attributs :
    - id_departement (INTEGER) : identifiant de l'Entité departement, est unique à l'Entité (PRIMARY KEY)
    - nom_departement (VARCHAR) : chaîne de caractères contenant le nom des différents nom des départements
    - id_region (INTEGER) : clé étrangère qui fait référence à l'Entité REGION. Il permet d'identifier la région du département où se trouve le festival.

- Entité COMMUNE : l'Entité commune contient les informations sur les communes du déroulement des festivals. Cette Entité compte quatre attributs :
    - code_insee (INTEGER) : identifiant de l'Entité commune, est aussi unique à l'Entité (PRIMARY KEY)
    - nom_commune (VARCHAR) : chaîne de caractères contenant le nom des différentes communes
    - code_postal (VARCHAR) : est le code postal de la commune
    - id_departement (INTEGER) : clé étrangère qui fait référence à l'Entité DEPARTEMENT. Il permet d'identifier le département de la commune où se déroule le festival.

- Entité DECENIE : l'Entité decenie contient les informations sur les décennies de création des différents festivals. Cette entité compte deux attributs :
    - id_decenie (INTEGER) : identifiant de l'entité decennie_creation qui est unique à l'entité (PRIMARY KEY)
    - nom_decenie (VARCHAR) : contient les décennies où les festivals ont été créés

- Entité PERIODE : l'entité contient les informations sur les dates de déroulement des festivals et est composée de deux attributs :
    - id_periode (INTEGER) : est l'identifiant de la date de déroulement , (PRIMARY KEY)
    - nom_periode (VARCHAR) : contient les périodes de déroulement

- Entité FESTIVAL : l'Entité festival contient les informations du festival. Cette Entité contient au total quatorze attributs :

    - id_festival (INTEGER) : identifiant de l'Entité festival, est aussi unique (PRIMARY KEY)
    - nom_festival (VARCHAR) : chaîne de caractères contenant les nom des festivals
    - site_internet (VARCHAR) : chaîne de caractères contenant les sites internets des festivals
    - adresse_mail (VARCHAR) : chaîne de caractères contenant les adresses mails des différents festivals
    - annee_creation (VARCHAR) : année de création des différents festivals
    - identifiant_agence (VARCHAR) : identifiant unique permettant de déterminer le code de l'agence
    - identifiant_cnm (VARCHAR) : identifiant unique permettant d'obtenir le du festival lier au Centre national de la musique
    - geocodage_x (VARCHAR) : coordonnées géographique horizontale (x)
    - geocodage_y (VARCHAR) : coordonnées géographique verticale (y)
    - id_envergure (INTEGER) : clé étrangère qui fait référence à l'Entité ENVERGURE permettant d'obtenir les informations concernant l'envergure du territoire des festivals
    - code_epci (INTEGER) : clé étrangère qui fait référence à l'Entité EPCI, permettant d'obtenir les informations des epci selon les festivals
    - code_insee (INTEGER) : clé étrangère qui fait référence à l'Entité COMMUNE, permettant d'obtenir les informations de la commune où se déroule les festivals
    - id_decenie (INTEGER) : clé étrangère qui fait référence à l'Entité DECENIE permettant d'obtenir les informations sur les décennies où les différents festivals ont été créés
    - id_periode (INTEGER) : clé étrangère qui fait référence à l'Entité PERIODE permettant d'obtenir les informations sur la période de déroulement des festivals

- Type Association ADRESSE_FESTIVAL : met en relation les Entités adresse et festival car un festival peut posséder plusieurs adresses de déroulement.
    - id_adresse (INTEGER) : identifiant de l'Entité adresse (clé étrangère)
    - id_festival (INTEGER) : identifiant de l'Entité festival (clé étrangère)
    - (id_festival et id_adresse) forme une clé primaire pour identifier chaque occurrence

- Type Association CATEGORIE_FESTIVAL : met en relation les Entités categorie et festival car un festival peut posséder plusieurs catégories.
    - id_categorie (INTEGER) : identifiant de l'Entité categorie (clé étrangère)
    - id_festival (INTEGER) : identifiant de l'Entité festival (clé étrangère)
    - (id_festival et id_categorie) forme une clé primaire pour identifier chaque occurrence

- Type Association SOUS_CATEGORIE_FESTIVAL : met en relation les Entités sous_categorie et festival car un festival peut posséder plusieurs sous_catégories.
    - id_sous_categorie (INTEGER) : identifiant de la Entité sous_categorie (clé étrangère)

    - id_festival (INTEGER) : identifiant de l'Entité festival (clé étrangère)
    - (id_festival et id_sous_categorie) forme une clé primaire pour identifier chaque occurrence

- Type Association ACTIVITE_FESTIVAL : met en relation les Entités activite et festival car un festival peut réaliser plusieurs activités.
    - id_activite (INTEGER) : identifiant de l'Entité activite (clé étrangère)
    - id_festival (INTEGER) : identifiant de l'Entité festival (clé étrangère)
    - (id_festival et id_activite) forme une clé primaire pour identifier chaque occurrence

- Type Association SOUS_CATEGORIE_ACTIVITE : met en relation les Entités sous_categorie et activite car une activité peut permettre de réaliser plusieurs sous catégories
    - id_activite (INTEGER) : identifiant de l'Entité activite (clé étrangère)
    - id_sous_categorie (INTEGER) : identifiant de l'Entité sous_categorie (clé étrangère)
    - (id_activite et id_sous_categorie) forme une clé primaire pour identifier chaque occurrence

- Type Association CATEGORIE_SOUS_CATEGORIE : met en relation les Entités sous_categorie et categorie car une catégorie peut contenir plusieurs sous catégories
    - id_sous_categorie (INTEGER) : identifiant de l'Entité sous_categorie (clé étrangère)
    - id_categorie (INTEGER) : identifiant de l'Entité categorie (clé étrangère)
    - (id_categorie et id_sous_categorie) forme une clé primaire pour identifier chaque occurrence.

Forme normale :

Dans notre modèle relationnel, nous avons donc l'ensemble des entités avec leurs attributs.

- Pour mettre ce modèle en **1NF**, il faut que chaque ligne de données des attributs soit atomique, c'est-à-dire que chaque ligne de données soit constituée uniquement d'une seule valeur. C'est pour cela que nous avons séparé le géocodage_xy en geocogade_x et geocodage_y car cette attributs n'était pas atomique avant cette étape. Cette étape est aussi le cas pour l'attribut des adresses telle que le numéro, nom,type de voie a part pour éviter le problème de multiplicité.

- Pour la **2NF**, il faut que les attributs de chaque entité soit en dépendances fonctionnelles avec la totalité de la clé de leur entité. C'est du coup, ce que nous avons réalisé avec notre modèle ER et EA car tous les attributs de chaque entité sont en dépendances fonctionnelles avec leur clé primaire.

- Pour la **3NF**, il ne faut pas qu'il y ait de dépendances transitives entre les attributs non clés. C'est ce que nous avons réalisé car dans notre modèle, nous avons pris soin de ne pas faire ce genre de relation entre les non clés des entités pour éviter les problèmes lors des créations de tables. Comme exemple, nous avons sorti code_epci et nom_epci dans une table seul.

- Pour la **BCNF**, il ne faut pas que nos schémas de relation contiennent des relation telle que les attributs non clés permettent de déterminer les attributs clés. Pour réaliser cela nous l'avons fait de manière à ne pas avoir d'erreur dans notre modèle relationnel. Ainsi, nous avons réalisé notre modèle et pour chaque entité, nous nous sommes interrogés sur s' il existait ce genre de dépendance dans notre modèle.

Pour conclure, nous avons pris pas mal de temps pour effectuer notre modèle relationnel et EA, pour vérifier toutes les conditions de normalisation et obtenir un modèle cohérent pour le jeu de données.

Explication de notre modèle

Nous avons effectué notre modèle EA (entité-association) et modèle relationnel de cette façon car nous avons interprété les choses de la manière suivante :

- Plusieurs adresses peut-être dans une ou plusieurs communes ainsi nous avons donc répartis les entités et nous les avons mis en lien avec une association.
- De plus, concernant les catégories et sous catégories, nous avions estimé que plusieurs sous catégories pouvaient appartenir à plusieurs catégories et inversement ainsi nous avons une relation (n,n) nous étions donc obligé d'ajouter une association entre les deux entités.
- Concernant les entités qui sont reliées au festival, nous les avons réalisées pour être sûr que chaque identifiant soit unique pour chaque festival (ce qui n'était pas le cas avant).
- Pour tout ce qui est zone géographique (région, département, commune) , nous nous somme basé sur une linéarité logique d'interprétation. C'est-à-dire qu' une commune permet de déterminer un département, et ce dernier permet de déterminer la région ainsi nous avons donc utilisé des clés étrangères pour faire le lien entre ces entités. L'inverse étant peu fiable car avoir le nom du département ne permet pas de connaître la commune ; nous avons répartis sous une forme de linéarité.
- Enfin, concernant les attributs de la base de données beaucoup de valeur des champs contenaient des lettres avec des chiffres nous avons donc utiliser des varchar sous peine d'avoir des problèmes dans notre base de données. Pour les identifiants, nous les avons mis en integer primary key (pour les clés primaire) et dans certaines entités nous retrouvons des integer foreign key (soit des clés étrangère faisant référence à une entité du modèle EA)

II. ETAPE 2 : Script de création et peuplement des tables de la base de données

1. Script SQL de création des tables.

```sql
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;
CREATE TABLE EPCI(
    code_epci INTEGER PRIMARY KEY,
    nom_epci VARCHAR
);

CREATE TABLE ENVERGURE(
    id_envergure INTEGER PRIMARY KEY,
    envergure_territorial VARCHAR
);

CREATE TABLE REGION(
    id_region INTEGER PRIMARY KEY,
    nom_region VARCHAR
);

CREATE TABLE DEPARTEMENT(
    id_departement INTEGER PRIMARY KEY,
    nom_departement VARCHAR,
    id_region INTEGER REFERENCES REGION(id_region) ON DELETE SET NULL
);

CREATE TABLE COMMUNE(
    code_insee INTEGER PRIMARY KEY,
    nom_commune VARCHAR,
    code_postal VARCHAR,
    id_departement INTEGER REFERENCES DEPARTEMENT(id_departement) ON
DELETE SET NULL
);

CREATE TABLE ADRESSE(
    id_adresse INTEGER PRIMARY KEY,
    num_voie VARCHAR,
    type_voie VARCHAR,
    nom_voie VARCHAR,
    complement_adresse VARCHAR,
    code_insee INTEGER REFERENCES COMMUNE(code_insee) ON DELETE SET NULL
);
```

```sql
CREATE TABLE CATEGORIE(
    id_categorie INTEGER PRIMARY KEY,
    nom_categorie VARCHAR
);

CREATE TABLE SOUS_CATEGORIE(
    id_sous_categorie INTEGER PRIMARY KEY,
    nom_sous_categorie VARCHAR
);

CREATE TABLE ACTIVITE(
    id_activite INTEGER PRIMARY KEY,
    nom_activite VARCHAR
);

CREATE TABLE DATE_DEROULEMENT (
    id_periode INTEGER PRIMARY KEY,
    nom_periode VARCHAR
);

CREATE TABLE DECENIE(
    id_decenie INTEGER PRIMARY KEY,
    nom_decenie VARCHAR
);

CREATE TABLE FESTIVAL(
    id_festival INTEGER PRIMARY KEY,
    nom_festival VARCHAR,
    site_internet VARCHAR,
    adresse_mail VARCHAR,
    annee_creation VARCHAR,
    identifiant_agence VARCHAR,
    identifiant_cnm VARCHAR,
    geocodage_x VARCHAR,
    geocodage_y VARCHAR,
    id_envergure INTEGER REFERENCES ENVERGURE(id_envergure) ON DELETE
SET NULL,
    code_epci INTEGER REFERENCES EPCI(code_epci) ON DELETE SET NULL,
    id_periode INTEGER REFERENCES DATE_DEROULEMENT(id_periode) ON DELETE
SET NULL,
    id_decenie INTEGER REFERENCES DECENIE(id_decenie) ON DELETE SET
NULL,
    code_insee INTEGER REFERENCES COMMUNE(code_insee) ON DELETE SET NULL
);
```

```sql
CREATE TABLE ADRESSE_FESTIVAL(
    id_adresse INTEGER REFERENCES ADRESSE(id_adresse) ON DELETE CASCADE,
    id_festival INTEGER REFERENCES FESTIVAL(id_festival) ON DELETE
CASCADE,
    PRIMARY KEY (id_adresse, id_festival)
);

CREATE TABLE CATEGORIE_FESTIVAL(
    id_categorie INTEGER REFERENCES CATEGORIE(id_categorie) ON DELETE
CASCADE,
    id_festival INTEGER REFERENCES FESTIVAL(id_festival) ON DELETE
CASCADE,
    PRIMARY KEY (id_categorie, id_festival)
);

CREATE TABLE SOUS_CATEGORIE_FESTIVAL(
    id_sous_categorie INTEGER REFERENCES
SOUS_CATEGORIE(id_sous_categorie) ON DELETE CASCADE,
    id_festival INTEGER REFERENCES FESTIVAL(id_festival) ON DELETE
CASCADE,
    PRIMARY KEY (id_sous_categorie, id_festival)
);

CREATE TABLE ACTIVITE_FESTIVAL(
    id_activite INTEGER REFERENCES ACTIVITE(id_activite) ON DELETE
CASCADE,
    id_festival INTEGER REFERENCES FESTIVAL(id_festival) ON DELETE
CASCADE,
    PRIMARY KEY (id_activite, id_festival)
);

CREATE TABLE SOUS_CATEGORIE_ACTIVITE(
    id_sous_categorie INTEGER REFERENCES
SOUS_CATEGORIE(id_sous_categorie) ON DELETE CASCADE,
    id_activite INTEGER REFERENCES ACTIVITE(id_activite) ON DELETE
CASCADE,
    PRIMARY KEY (id_sous_categorie, id_activite)
);
CREATE TABLE CATEGORIE_SOUS_CATEGORIE(
    id_sous_categorie INTEGER REFERENCES
SOUS_CATEGORIE(id_sous_categorie) ON DELETE CASCADE,
    id_categorie INTEGER REFERENCES CATEGORIE(id_categorie) ON DELETE
CASCADE,
    PRIMARY KEY (id_categorie, id_sous_categorie)
);
```

2. Script SQL de peuplement des tables.

a. code jupyter

Nous sommes partis du fichier CSV à plat contenant toutes les données. Pour répondre aux exigences de normalisation de notre base de données, nous avons rédigé un script Python utilisant la bibliothèque pandas.

Ce script a permis de transformer et de scinder le fichier CSV de départ en plusieurs fichiers distincts, chacun respectant les règles de normalisation de la BCNF et étant structuré comme la base de données que nous avons créé précédemment. En utilisant pandas, nous avons pu manipuler les données facilement.

Enfin, une fois les fichiers normalisés générés, nous avons utilisé la commande SQL \copy pour importer les données dans la base de données. Cette approche nous a assuré que les données étaient correctement organisées, facilitant ainsi leur intégration et utilisation future.

NB : nous vous fournirons le notebook contenant le code python intégral

b. code sql

```
\i E:/thierno/TRAVAIL/S2_SQL/sae201/gendb.sql

\encoding utf8
\copy EPCI FROM E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/epci.txt
\copy ENVERGURE FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/envergure.txt
\copy REGION FROM E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/region.txt
\copy departement FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/departement.txt
\copy commune FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/commune.txt
\copy adresse FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/adresse.txt
\copy categorie FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/categorie.txt
\copy sous_categorie FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/sous_categorie.txt
\copy activite FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/activite.txt
\copy date_deroulement FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/periode.txt
\copy decenie FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/decennie.txt
```

```
\copy festival FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/festival.txt
\copy adresse_festival FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/adresse_festival.txt
\copy categorie_festival FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/categorie_festival.txt
\copy sous_categorie_festival FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/sous_categorie_festival.txt
\copy activite_festival FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/activite_festival.txt
\copy categorie_sous_categorie FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/sous_categorie_categorie.txt
\copy sous_categorie_activite FROM
E:/thierno/TRAVAIL/S2_SQL/sae201/peuplement/sous_categorie_activite.txt
```

3. Description commentée des différentes étapes du script de peuplement.

Explication des scripts : Tout d'abord, ce qu'il faut savoir est que nous avons réalisé nos scripts de création des tables en fonction de notre modèle EA et pour notre script de peuplement des tables nous avons utilisé Python pour remplir l'ensemble des tables du jeu de données. C'est-à-dire que tout d'abord, la répartition des données dans les tables étaient nécessaires. Cependant ce point présentait plusieurs soucis telle que les données de certaines colonnes possédaient des caractères différents (chiffres, lettres, etc). L'objectif en premier a été de tout mettre en norme. Ceci fait, nous avons donc rempli les entités de notre modèle avec les données du fichier. Enfin, pour intégrer l'ensemble des entités possédant toutes les données à la base de données créée, nous avons recouru à la fonction SQL : \copy afin d'avoir tout ce qu'il nous faut pour la suite, c'est-à-dire l'interrogation de la base de données à l'aide des requêtes.

III. ETAPE 3 : Interrogation de la base de données et visualisation des résultats

1. Interrogation variée de la base de données.

La base de données présente plusieurs opportunités d'analyses et donc nous voulons savoir la répartition des festivals selon les catégories, les sous catégories, les régions, selon les décennies de création et les départements. Pour ce faire, nous allons aborder différentes requêtes SQL afin d'avoir une approche plus visuelle de ce que nous voulons savoir et en tirer des conclusions.

2. Script SQL concernant les requêtes.

Requête 1 : Catégorie dominante

```
SELECT c.nom_categorie, COUNT(cf.id_festival) AS nombre_festivals
FROM CATEGORIE c
JOIN CATEGORIE_FESTIVAL cf USING (id_categorie)
GROUP BY c.nom_categorie
ORDER BY nombre_festivals DESC;
```

Requête 2 : Répartition des festival par décennie de création

```
SELECT d.nom_decenie, COUNT(f.id_festival) AS nombre_festivals
FROM FESTIVAL f
JOIN DECENIE d USING (id_decenie)
GROUP BY d.nom_decenie
ORDER BY nombre_festivals DESC;
```

Requête 3 : Nombre de festival par région

```
SELECT r.nom_region, COUNT(f.id_festival) AS nombre_festivals
FROM FESTIVAL f
JOIN COMMUNE c USING (code_insee)
JOIN DEPARTEMENT d USING(id_departement)
JOIN REGION r USING(id_region)
GROUP BY r.nom_region
ORDER BY nombre_festivals DESC;
```

Requête 4 : Répartition des festivals par sous catégorie

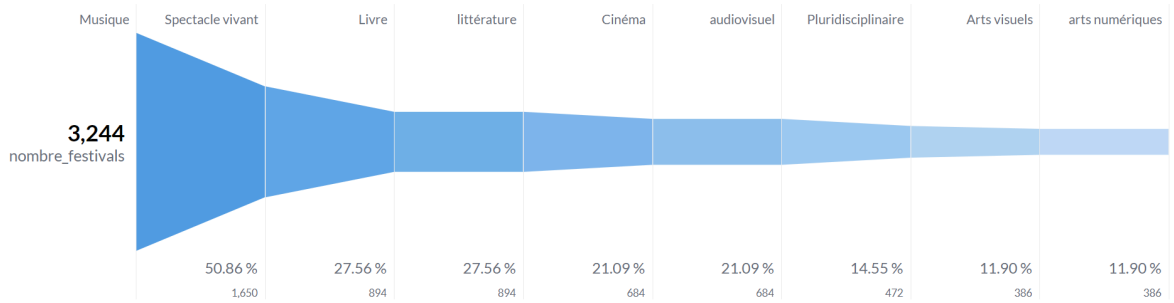
```
SELECT COUNT(nom_festival) AS c, nom_sous_categorie
FROM FESTIVAL
JOIN SOUS_CATEGORIE_FESTIVAL USING (id_festival)
JOIN SOUS_CATEGORIE USING (id_sous_categorie)
GROUP BY nom_sous_categorie ORDER BY c DESC ;
```

Requête 5 : nombre de festivals par département

```
SELECT count(*) e, nom_departement
FROM festival
JOIN commune using(code_insee)
JOIN departement using(id_departement)
GROUP BY nom_departement;
```

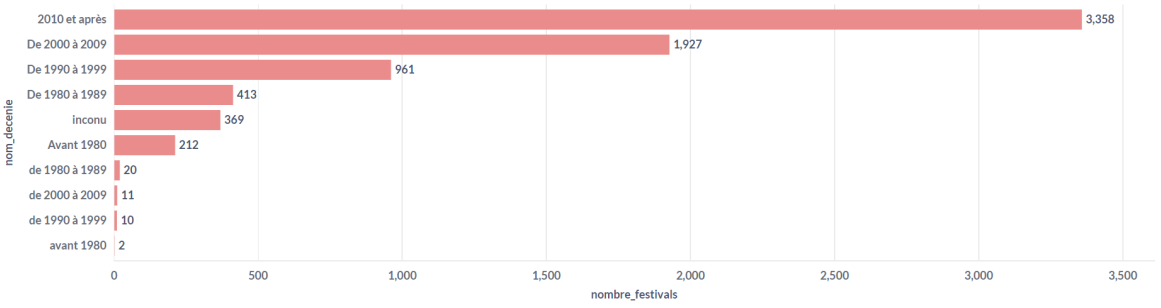
3. Visualisation et commentaire des résultats

Requête 1 : Catégorie dominante



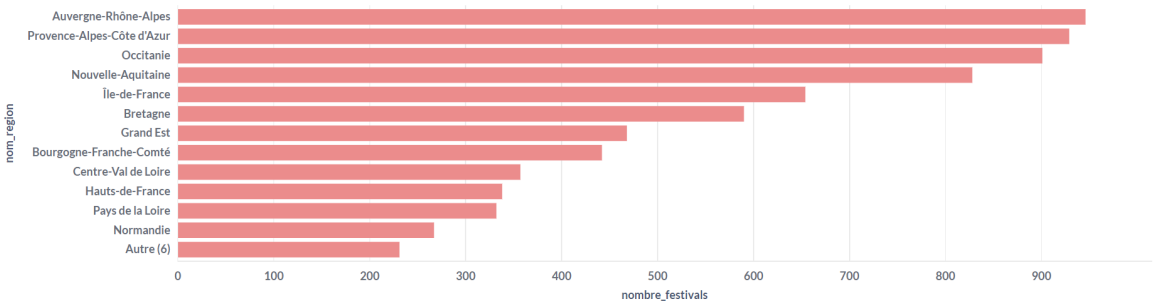
Ce graphique est un entonnoir, qui compte le nombre de festivals pour chaque catégorie. Nous observons que la catégorie musique avec un total de 3244 festivals est la plus importante suivie par les catégories spectacle vivant et livre. Nous observons aussi que la somme des pourcentages est supérieure à 100%, ce qui veut dire qu'il y a des festivals qui comptent plusieurs catégories. Ce qui est cohérent car beaucoup de festivals sont des festivals de musique qui ont souvent lieu en été.

Requête 2 : Répartition des festival par décennie de création



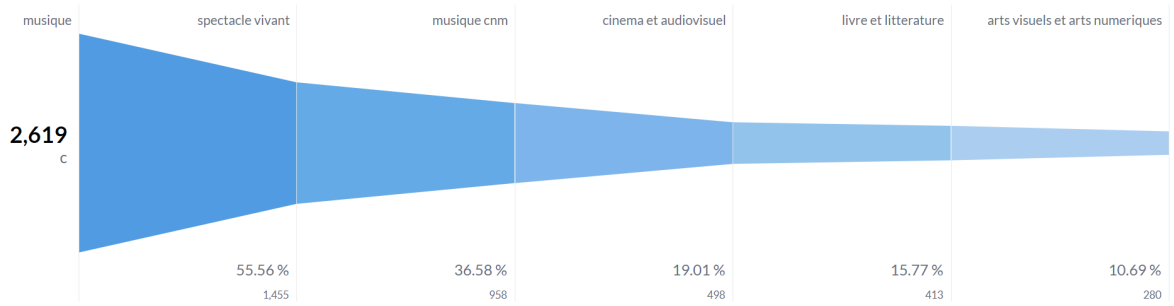
Pour cette visualisation, nous pouvons voir qu'il s'agit d'un diagramme en bar qui a comme sélection le nombre de festivals par décennie. C'est-à-dire que pour une décennie, nous avons un tel nombre de festivals. Comme par exemple,
Pour la décennie de 2000 à 2009, il y a au total 1927 festivals. Ce cas est expliqué par l'invention de plusieurs thèmes et l'invention notamment de plusieurs festivals dans les années 2000 et ce confirme par la tendance croissante des créations de festivals pour les décennies surtout à partir des années 80.

Requête 3 : Nombre de festival par région



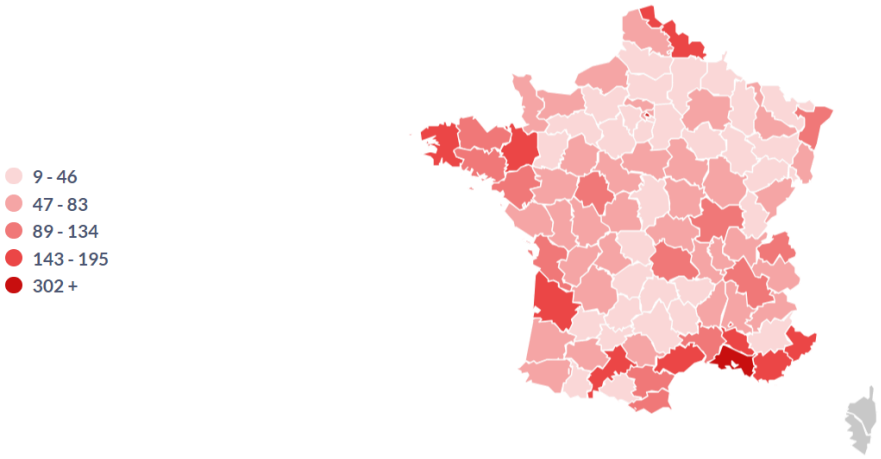
Pour cette visualisation, nous avons la répartition du nombre de festivals situés dans les différentes régions de France. A travers ce graphique, nous pouvons apercevoir que l'Auvergne-Rhône-Alpes, Provence-Alpes-Côte d'Azur et autres ont davantages de festivals. Ce phénomène est dû notamment à la localisation de ces régions situées principalement au Sud, Sud-Est notamment car dans le Sud, il y a le soleil, la bonne humeur, etc, c'est pour cela qu'il y a davantage de festivals dans ces zones géographiques.

Requête 4 : Répartition des festivals par sous catégorie.



Pour la visualisation ci-dessus, il s'agit à nouveau d'un entonnoir qui compte le nombre de festivals pour chaque sous catégories. Nous apercevons une répartition majeure pour musique avec un total de 2619 festivals suivi de spectacle vivant (1455 festivals). Cette représentation est dû à une forte présence de la musique en France que ce soit concerts ou festivals en général et c'est pour cela qu'il est davantage représenté.

Requête 5 : Nombre de festivals par département



Pour cette visualisation, il s'agit d'une répartition du nombre de festivals par département visualisée par sur une carte de France. D'après la visualisation, nous constatons qu'il y a un dégradé de couleurs pour montrer les différences entre les départements. C'est-à-dire que chaque département ayant une couleur plus foncée montre qu'il y a beaucoup plus de festivals dans ces zones que dans les autres. Enfin, nous remarquons que les départements qui ont le plus de festivals se situent davantage vers les côtes du fait que c'est plus attirant pour assister aux festivals qui sont proches de la mer, du soleil. Ce qui rejoint le constat que nous avons fait avec les régions.

CONCLUSION

Pour conclure cette SAE, nous avons été obligés de recourir à nos connaissances théoriques reçues durant le semestre 2 en passant par la normalisation, les scripts SQL, ainsi que les requêtes, nous avons pu effectuer les tâches souhaitées. De plus, il nous fallait préparer notre travail pour l'oral qui avait eu lieu pour cela nous avons recouru à notre apprentissage en communication, pour présenter notre travail.