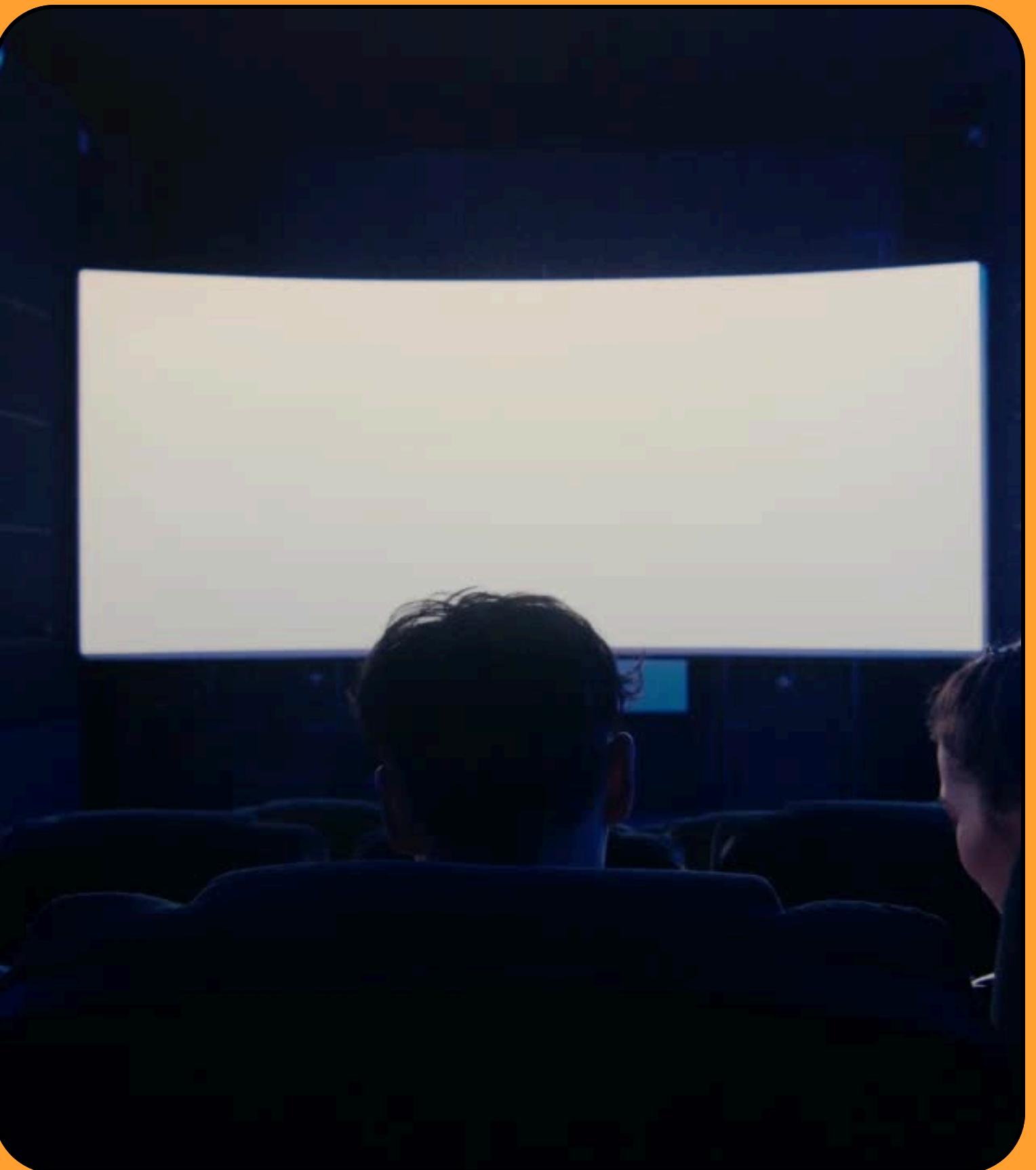
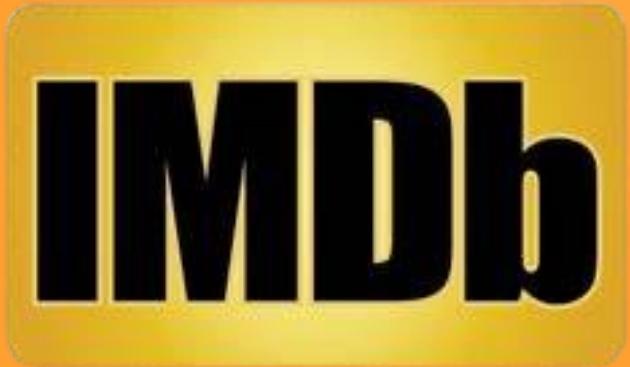
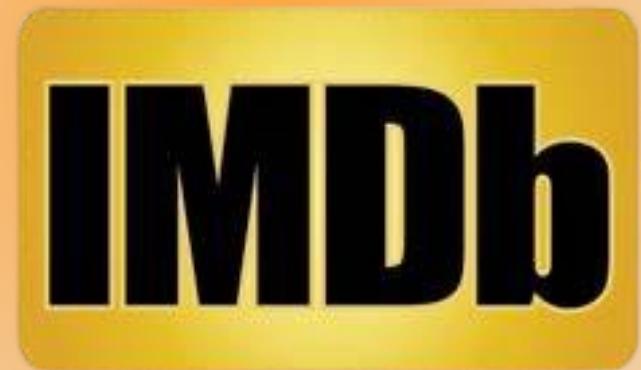


Soutenance du Projet **IMDb**



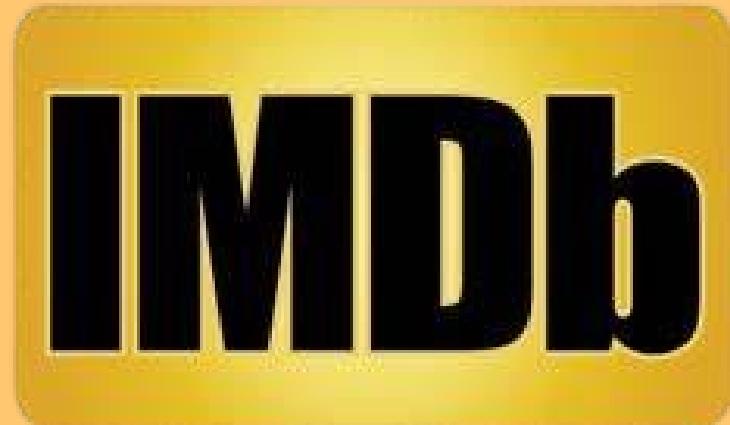
Vue d'ensemble



- 01**
Introduction et Contexte
- 02**
Besoins fonctionnels globaux
- 03**
Conception de la Base OLTP
- 04**
Optimisations de la Base OLTP
- 05**
Importation des Données
- 06**
Conception de la Base OLAP et Power BI
- 07**
Modèles IA/Bag of words
- 08**
Gantt
- 09**
Conclusion

01

Introduction et Contexte



Introduction et Contexte

Contexte :

- IMDb : Base de données mondiale sur le cinéma et la télévision.
- l'un des sites les plus visités au monde
- Son accès est gratuit pour les données publiques



```
3 require File.expand_path("../../.env", __FILE__)
4 # Prevent database truncation if the environment is test
5 abort("The Rails environment is running in production mode") if Rails.env == "production"
6 require 'spec_helper'
7 require 'rspec/rails'

8 require 'capybara/rspec'
9 require 'capybara/rails'

10 Capybara.javascript_driver = :webkit
11 Category.delete_all; Category.create!(name: "Science Fiction")
12 Shoulda::Matchers.configure do |config|
13   config.integrate do |with|
14     with.test_framework :rspec
15     with.library :rails
16   end
17 end
18
19 # Add additional requires below this line to append them to all features
20
21 # Requires supporting ruby files with custom matchers and helpers
22 # in spec/support/ and its subdirectories. This file is normally loaded
23 # in _spec.rb will both be required after those files.
24 # run as spec files by default. You can change this using
25 # --require spec/support/integration.rb
26 # run twice. It is recommended that you do not name
27 # end with _spec.rb. You can configure the
28 # location of this file in your
29 # configuration on the command line or in
30 # .rspec file. If you do not specify a language
31 # it is taken to be "ruby"
32
33 # mongoid
34 # buffer
```

02

Analyse des Besoins

[Retourner vers la vue d'ensemble](#)



Analyse des Besoins

Rapports et analyse de données :

- Navigation et visualisation avec Power BI.
- Extraction de tendances et insights.

Modèles IA :

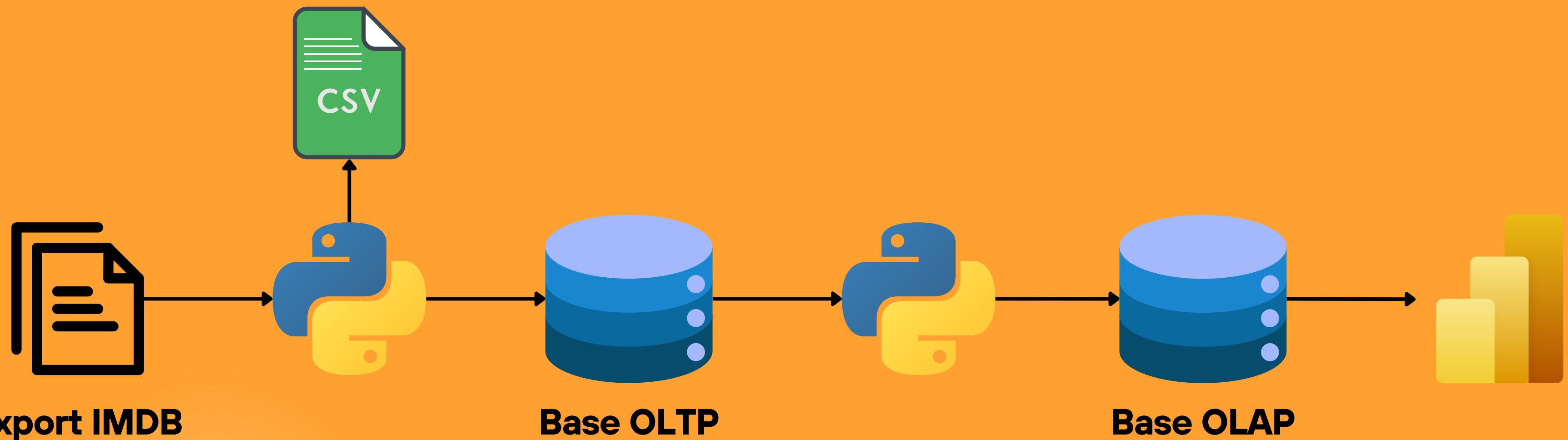
- Classification des sentiments.
- Attribution de notes et scores pour les critiques.

Gestion de projet :

- Planification (GANTT) :

Introduction

Flux des données



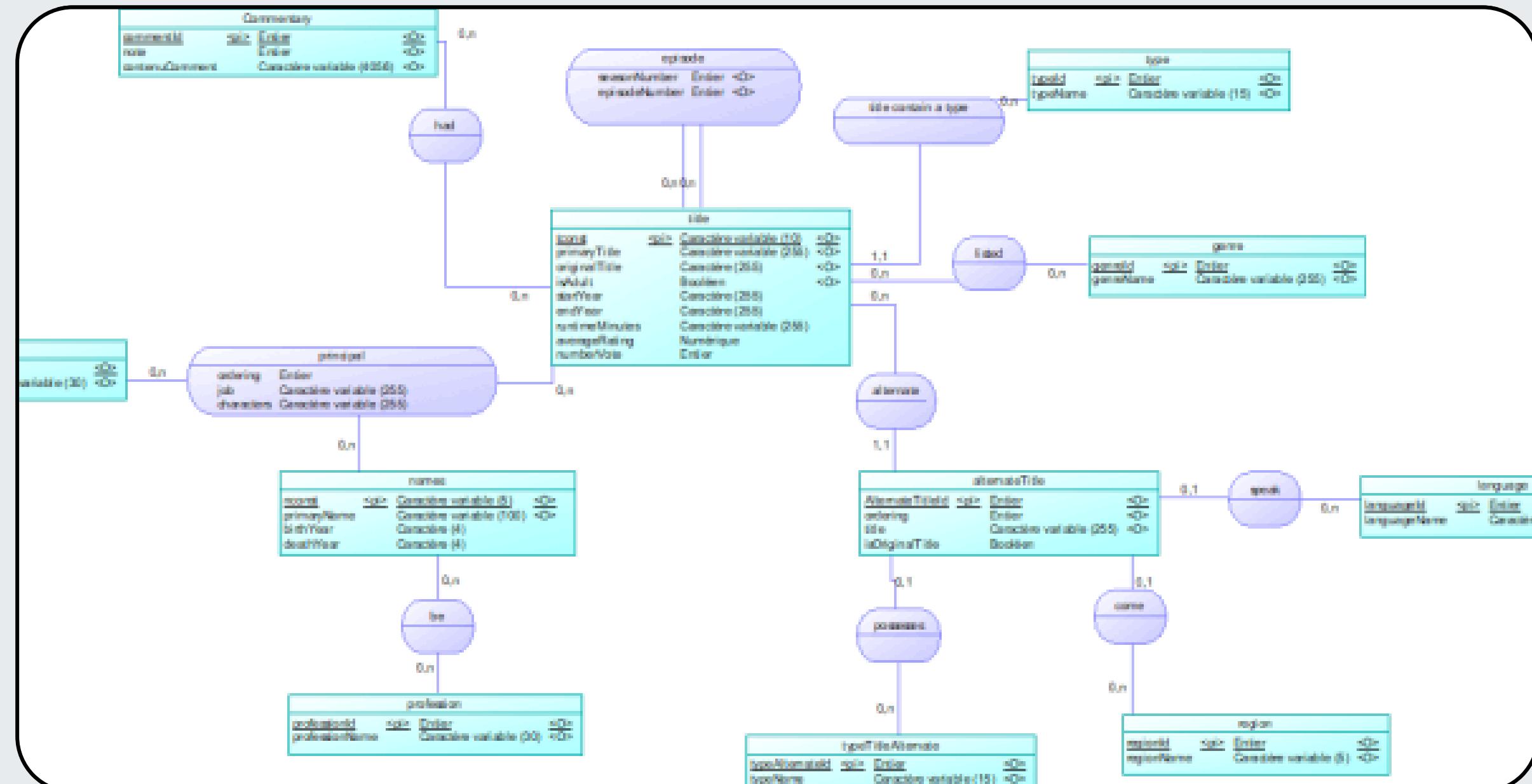
[Retourner vers la vue d'ensemble](#)

02

Conception de la Base OLTP

[Retourner vers la vue d'ensemble](#)

Conception de la Base OLTP

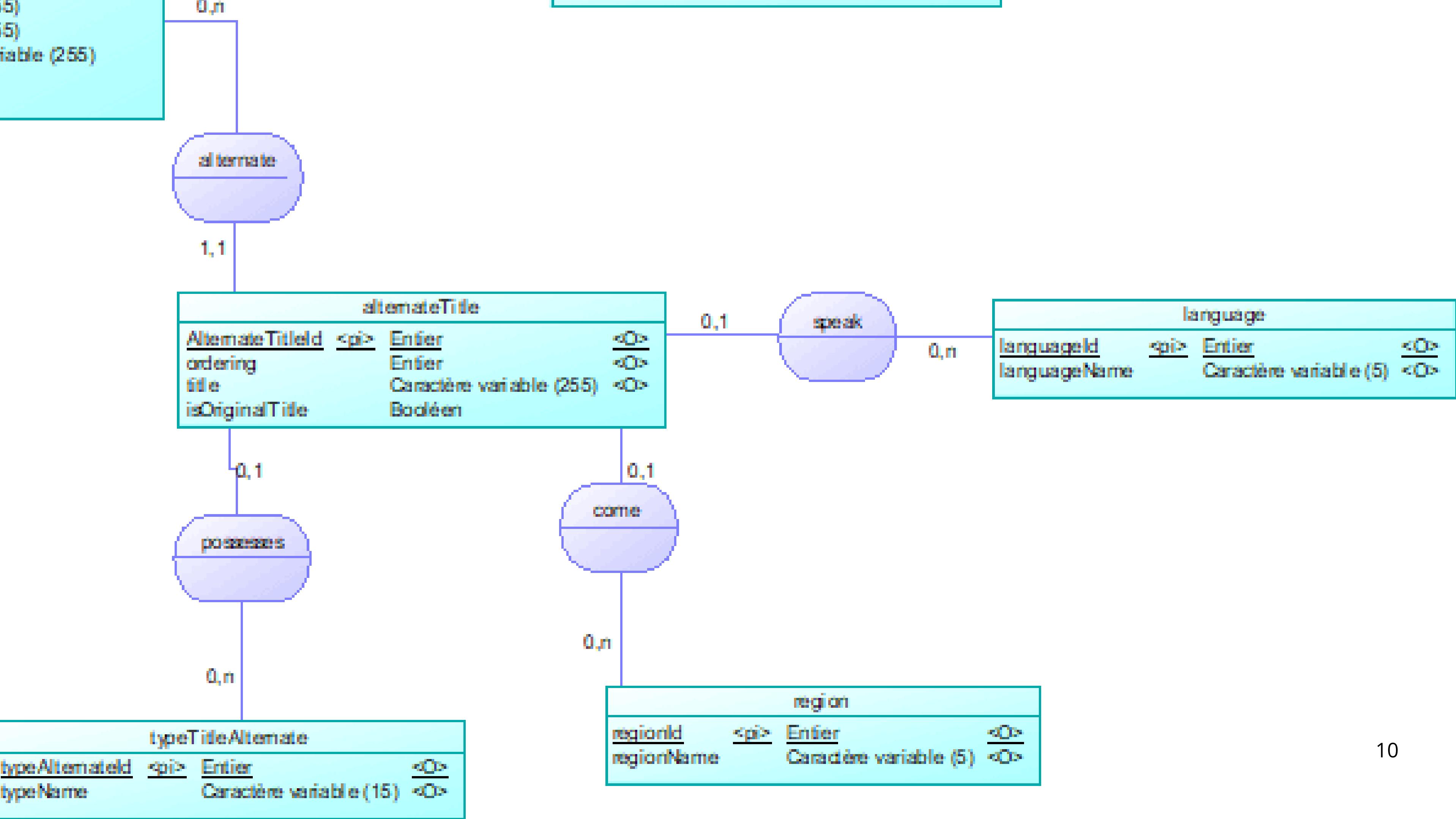


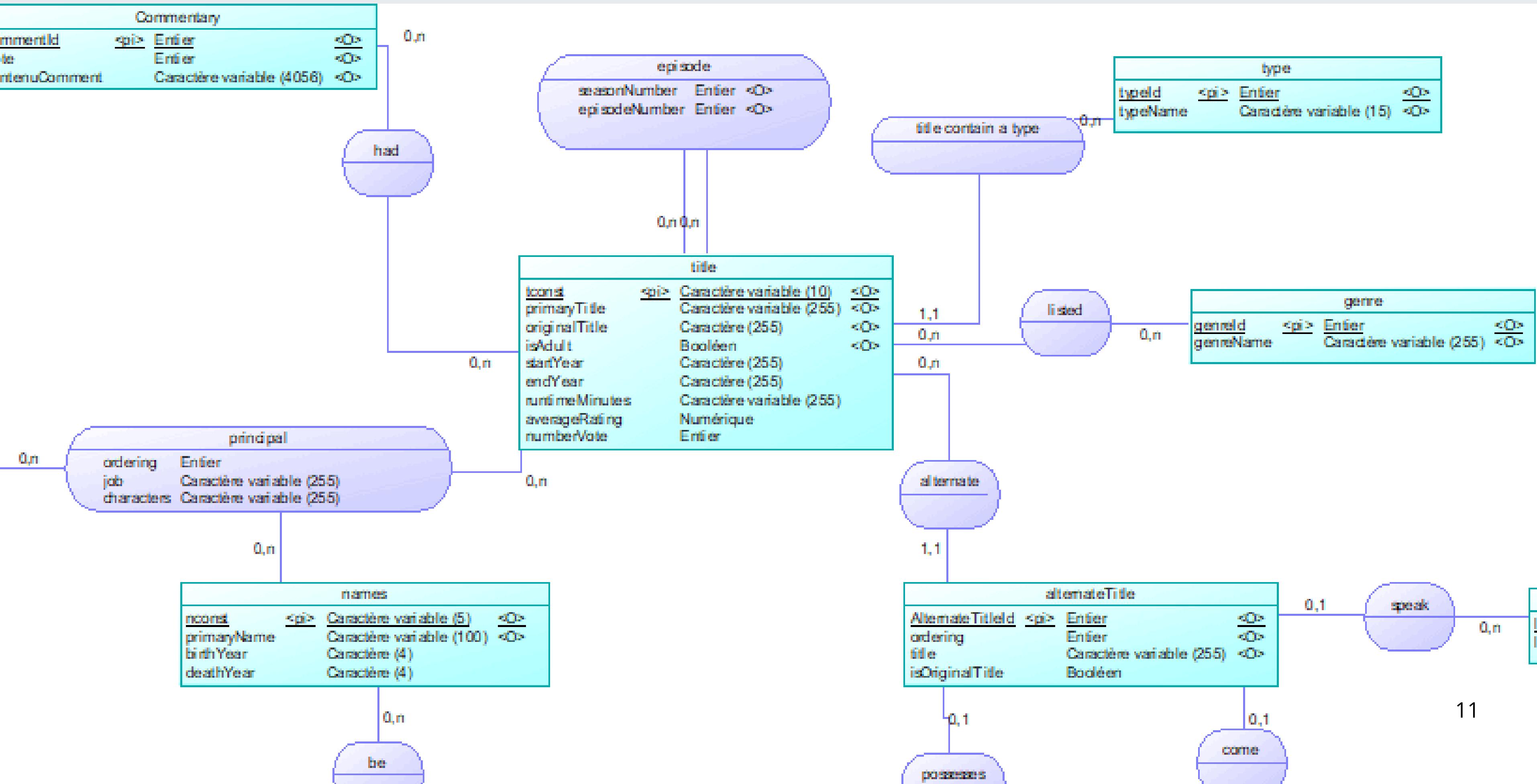
Schémas de conception :

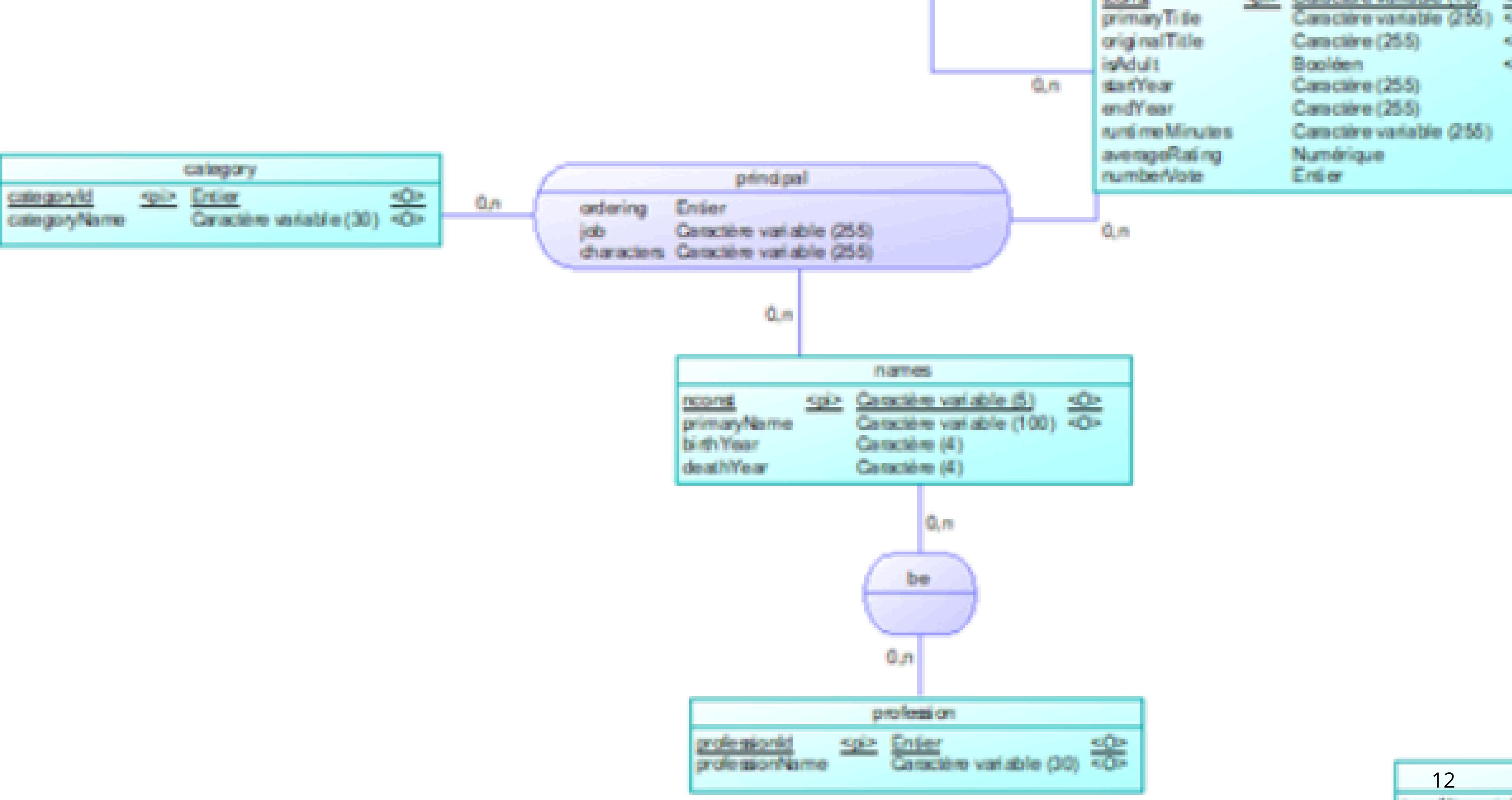
- Modèle Conceptuel de Données (MCD).
- Modèle Logique de Données (MLD).
- Modèle Physique de Données (MPD).

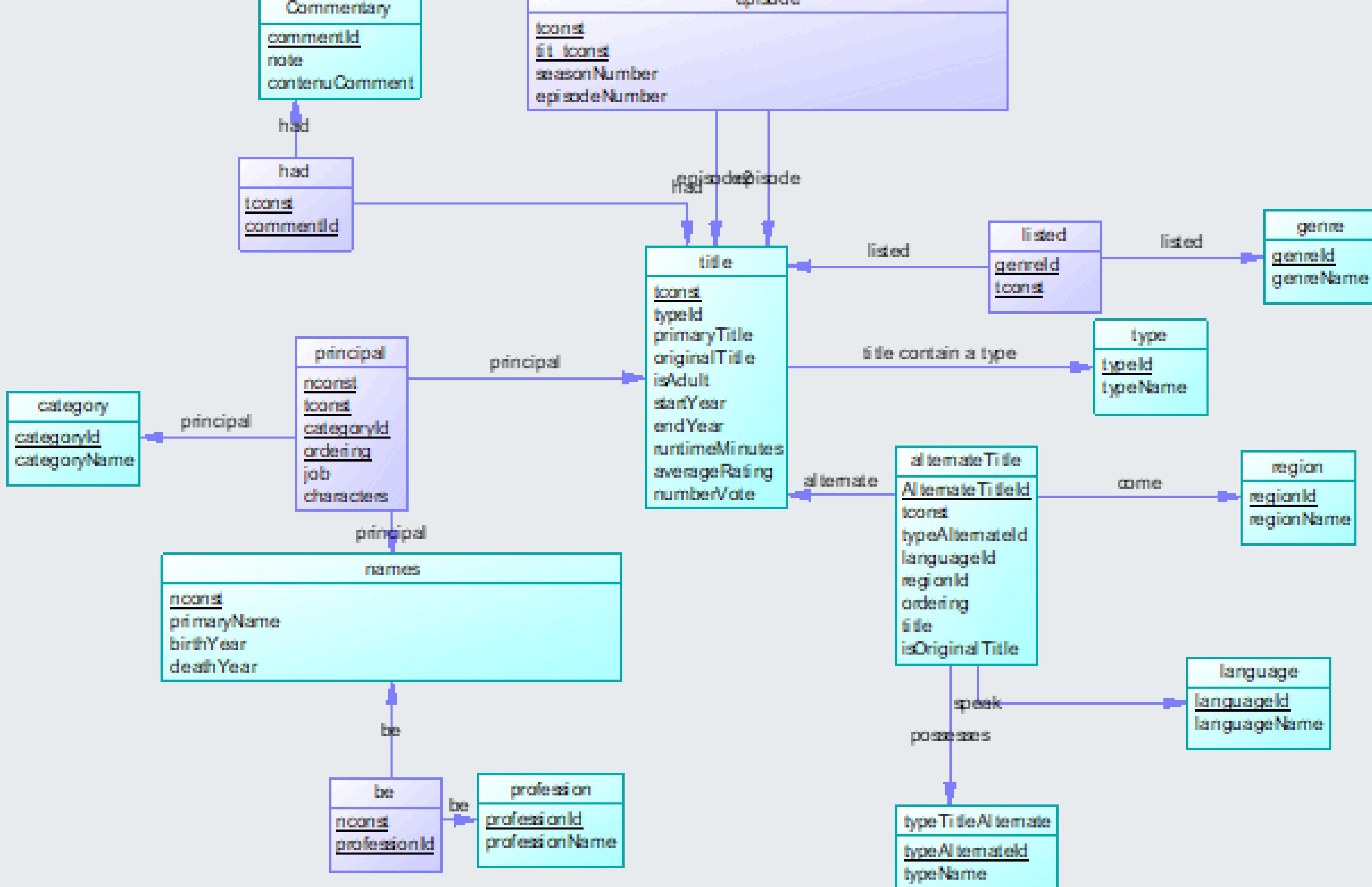
Choix de conception :

- Normalisation pour réduire la redondance.
- Gestion efficace des relations entre entités.









03

Optimisations OLTP

Caractéristique clé 1

Techniques appliquées :

- Indexation

Résultats :

- Réduction des temps de requêtes.



Indexation

Index sur les titres et films

primary title ; averagerating ; numbervote

Index sur les personnes

primaryname ; birthyear

Index sur les critiques

note

Index sur les relations

title ; tconst ; categoryid

04

Importation des Données

Importation des Données



```
import pandas as pd  
import time  
import pyodbc  
  
from sqlalchemy import create_engine
```

Source :

- IMDb et jeu de données Stanford.

Méthodologie :

- Outils utilisés : Python.
- Volumétrie des données : 25 000 critiques pour entraînement et test.

Défis rencontrés :

- Gestion de la volumétrie.
- Nettoyage des données.

Importation des Données

```
# 🔒 Connexion SQL Server
server = "193.48.129.35,784"
username = "AGED23_G3B"
password = "Av9HAWNV"

OLTP_DB_URL = f"mssql+pyodbc://{{username}}:{{password}}@{{server}}/AGED23_G3B?driver=ODBC+Driver+17+for+SQL+Server&TrustServerCertificate=yes"
OLAP_DB_URL = f"mssql+pyodbc://AGED23_G3A:{{password}}@{{server}}/AGED23_G3A?driver=ODBC+Driver+17+for+SQL+Server&TrustServerCertificate=yes"

# Création des connexions aux bases OLTP et OLAP
engine.oltp = create_engine(OLTP_DB_URL)
engine.olap = create_engine(OLAP_DB_URL)

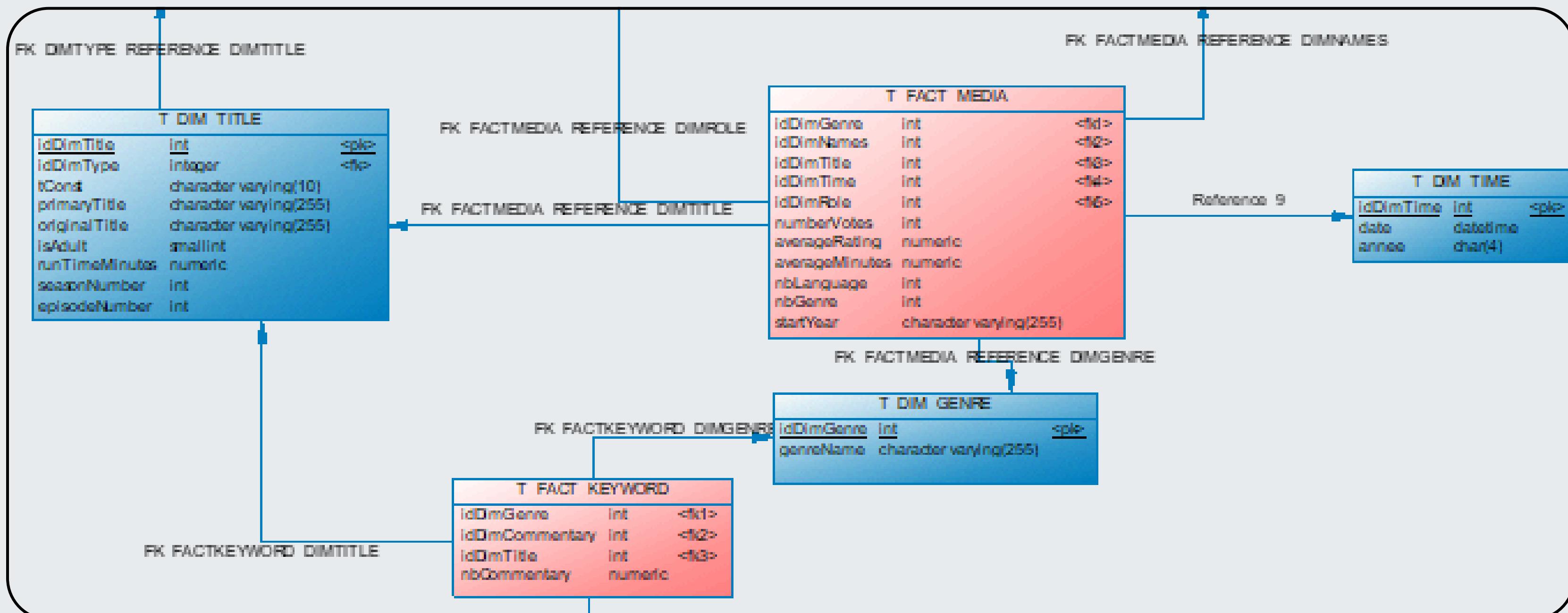
# 📁 Extraction des données depuis OLTP
def extract_names():
    print("📥 Extraction des données de NAMES...")
    start_time = time.time()
    query = "SELECT DISTINCT NCONST, PRIMARYNAME, BIRTHYEAR, DEATHYEAR FROM NAMES"
    df = pd.read_sql(query, engine.oltp)
    print(f"✅ Extraction terminée en {time.time() - start_time:.2f} sec")
    return df
```

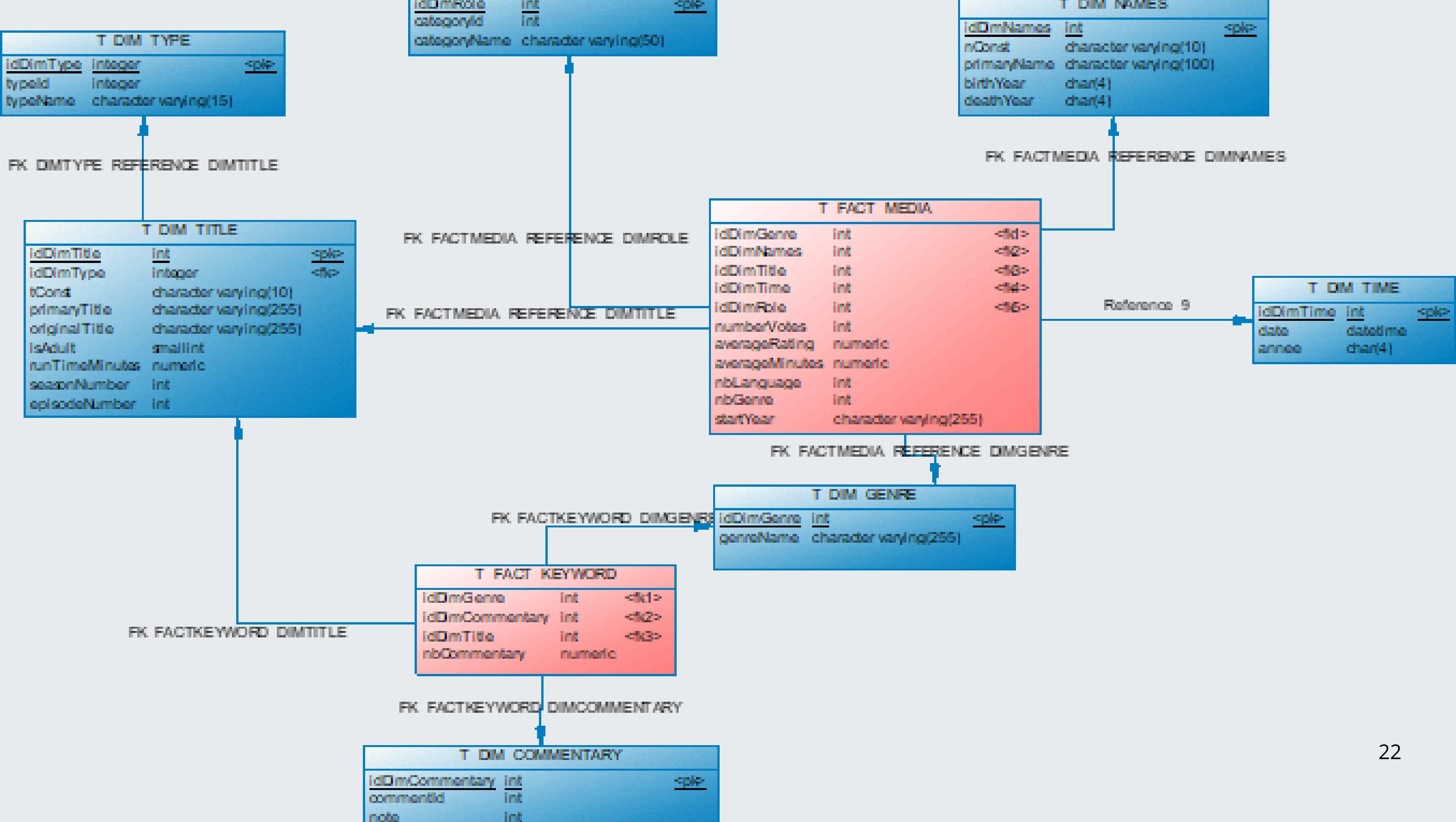
```
✓ def insert_names_batch(df, batch_size=10000):
```

06

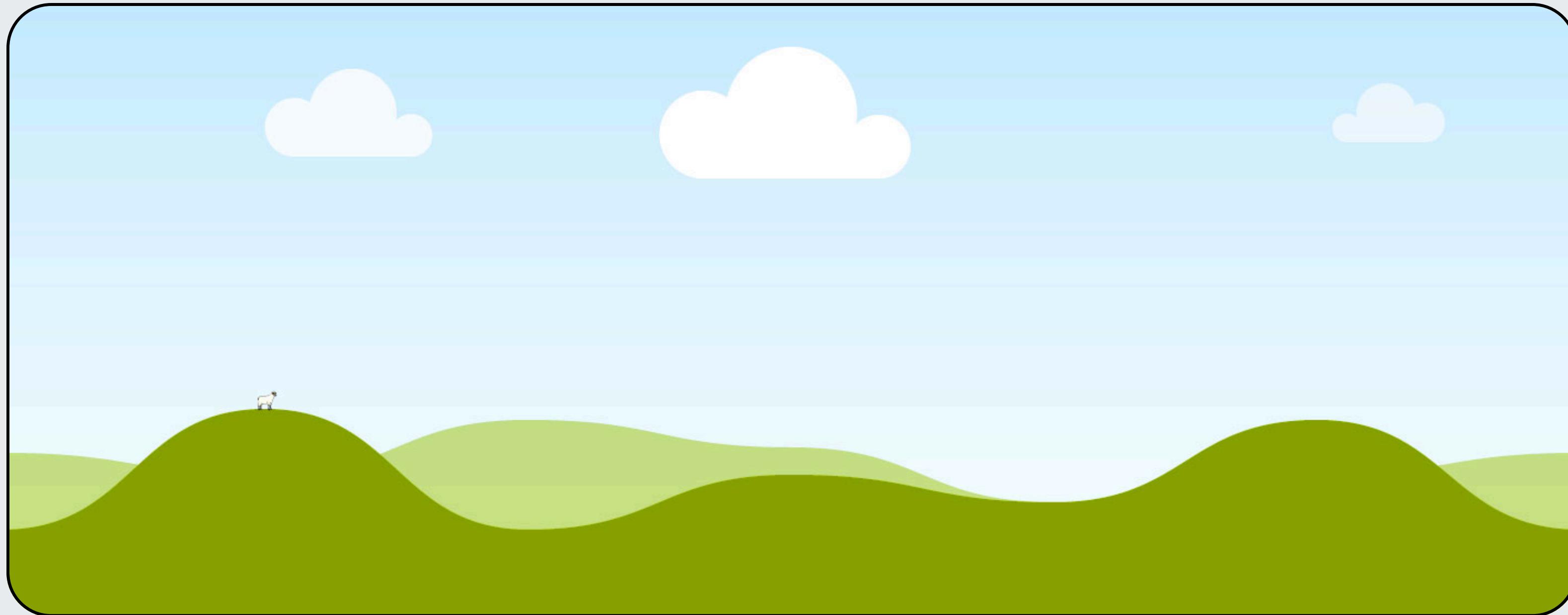
Conception de la Base OLAP et Power BI

Conception de la Base OLAP





Rapports Power BI



07

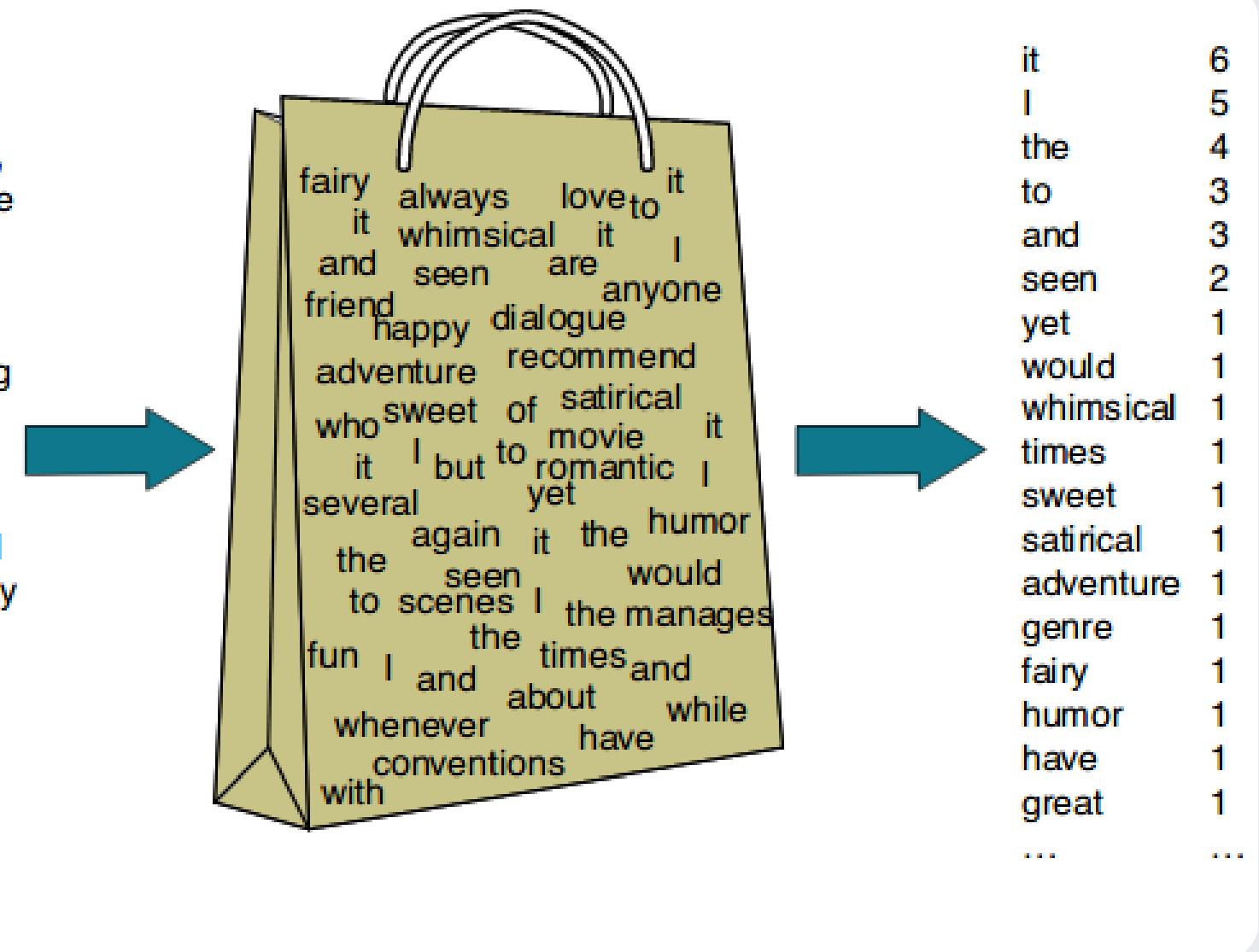
Modèles IA/Bag of words

Bag of words

Explication :

BOW est une technique de représentation textuelle en NLP qui convertit un texte en une matrice numérique

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



Récupération et nettoyage des données

 test
 train

 neg
 pos
 unsup

 0_9.txt
 1_7.txt
 2_9.txt
 3_10.txt
 4_8.txt
 5_10.txt
 6_10.txt
 7_7.txt
 8_7.txt
 9_7.txt
 10_9.txt

	review	sentiment
0	warm touch movi fantasylik qualityellen bursty...	positive
1	anyon find film bore hopeless bonehead stick c...	positive
2	certainli without merit alreadi writerdirector...	negative
3	immor tale left bad tast mouth seem borowczyk ...	negative
4	watch entir movi recogn particip william hurt ...	negative

```
# Création de la datadrame à partir du dossier contenant tous les commentaires, pour le dossier train et pour le dossier test
def create_dataframe_from_folders(pos_dir, neg_dir):
    data = []

    # Process du dossier 'pos'
    for filename in os.listdir(pos_dir):
        if filename.endswith(".txt"):      # On vérifie qu'il s'agit bien d'un fichier .txt, format des commentaires à récupérer
            file_path = os.path.join(pos_dir, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                content = file.read().strip()
                # On récupère le commentaire dans le fichier texte, et on lui associe le sentiment positif qu'on connaît déjà
                data.append({"review": content, "sentiment": "positive"})

    # Process du dossier 'neg', idem que le dossier pos
    for filename in os.listdir(neg_dir):
        if filename.endswith(".txt"):
            file_path = os.path.join(neg_dir, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                content = file.read().strip()
                data.append({"review": content, "sentiment": "negative"})
                # De la même manière on associe le sentiment négatif ici

    # Création de la dataframe globale
    df = pd.DataFrame(data)

    # Randomisation de l'ordre de la dataframe pour que l'algorithme n'apprenne pas d'abord que des positifs puis des négatifs
    df = df.sample(frac=1).reset_index(drop=True)

    return df
```

```
def remove_html(text):
    bs = BeautifulSoup(text, "html.parser")
    return ' ' + bs.get_text() + ' '
def keep_only_letters(text):
    text=re.sub(r'^[^\w\s]', '', text)
    return text
def convert_to_lowercase(text):
    return text.lower()
def clean_reviews(text):
    text = remove_html(text)
    text = keep_only_letters(text)
    text = convert_to_lowercase(text)
    return text
```

```
# Téléchargement des stopwords de nltk
nltk.download('stopwords')

english_stop_words = nltk.corpus.stopwords.words('english')

# Fonction permettant de retirer les mots vides du texte
def remove_stop_words(text):
    for stopword in english_stop_words:
        stopword = ' ' + stopword + ' '
        text = text.replace(stopword, ' ')
    return text
```

1e Modèle IA : Sentiment

Modèle Sentiment :
Prédire si le contenu du commentaire est négatif ou positif

```
vectorizer = CountVectorizer(binary=False,ngram_range=(1,2), min_df=0.01)
tf_features_train = vectorizer.fit_transform(imdb_train['review'])
tf_features_test = vectorizer.transform(imdb_test['review'])
print ('Dimensions du Bag of Words : ',tf_features_train.shape)
clf = LinearSVC(C=0.1)
clf.fit(tf_features_train, train_labels)
predictions = clf.predict(tf_features_test)
print(classification_report(test_labels, predictions, target_names=['Negative', 'Positive']))
print(confusion_matrix(test_labels, predictions, labels=[0, 1]))
```

	Dimensions du Bag of Words : (25000, 1745)			
	precision	recall	f1-score	support
Negative	0.87	0.85	0.86	12500
Positive	0.86	0.87	0.86	12500

2e Modèle IA : Notes

Modèle Note :

Prédire une note entre 1 et 4 ou 7 et 10 en analysant le texte du commentaire

Confusion Matrix:

```
[[3703  295  263  298   84   54   40  285]
 [1219  225  212  293   86   75   33  159]
 [ 885  288  349  452  170  119   45  233]
 [ 604  218  354  652  265  189   86  267]
 [ 153   69  112  267  497  445  186  578]
 [ 162   44   90  211  435  495  296 1117]
 [ 132   32   46  100  225  342  234 1233]
 [ 252   43   77  114  238  484  368 3431]]
```

```
vectorizer = CountVectorizer(binary=False, ngram_range=(1, 2), min_df=0.01)
# On applique ce Bag of Words à notre jeu d'entraînement et notre jeu de test.
tf_features_train = vectorizer.fit_transform(imdb_train['review'])
tf_features_test = vectorizer.transform(imdb_test['review'])
print('Bag of Words créé correctement')

# Afficher les dimensions du BoW
print(f"Dimensions du BoW d'entraînement : {tf_features_train.shape}")
print(f"Dimensions du BoW de test : {tf_features_test.shape}")

# Création de notre modèle d'IA, on utilise LinearSVC
clf = LinearSVC(C=0.1)
# Entraînement de notre modèle sur le jeu d'entraînement créé au préalable
clf.fit(tf_features_train, imdb_train['rating'])

# Ajout d'un timer de progression pendant la prédiction
print("\nPredictions en cours...")
predictions = []
for i in tqdm(range(tf_features_test.shape[0]), desc="Prédiction", unit="échantillon"):
    predictions.append(clf.predict(tf_features_test[i])) # Passer directement l'échantillon sans la liste
predictions = np.array(predictions).flatten() # Convertir la liste de prédictions en un tableau
```

2e Modèle IA : Notes

```
def group_ratings(rating):
    if rating in [1, 2]:
        return '1-2'
    elif rating in [3, 4]:
        return '3-4'
    elif rating in [7, 8]:
        return '7-8'
    elif rating in [9, 10]:
        return '9-10'
    else:
        return 'Unknown' # Au cas où il y aurait des notes non prévues

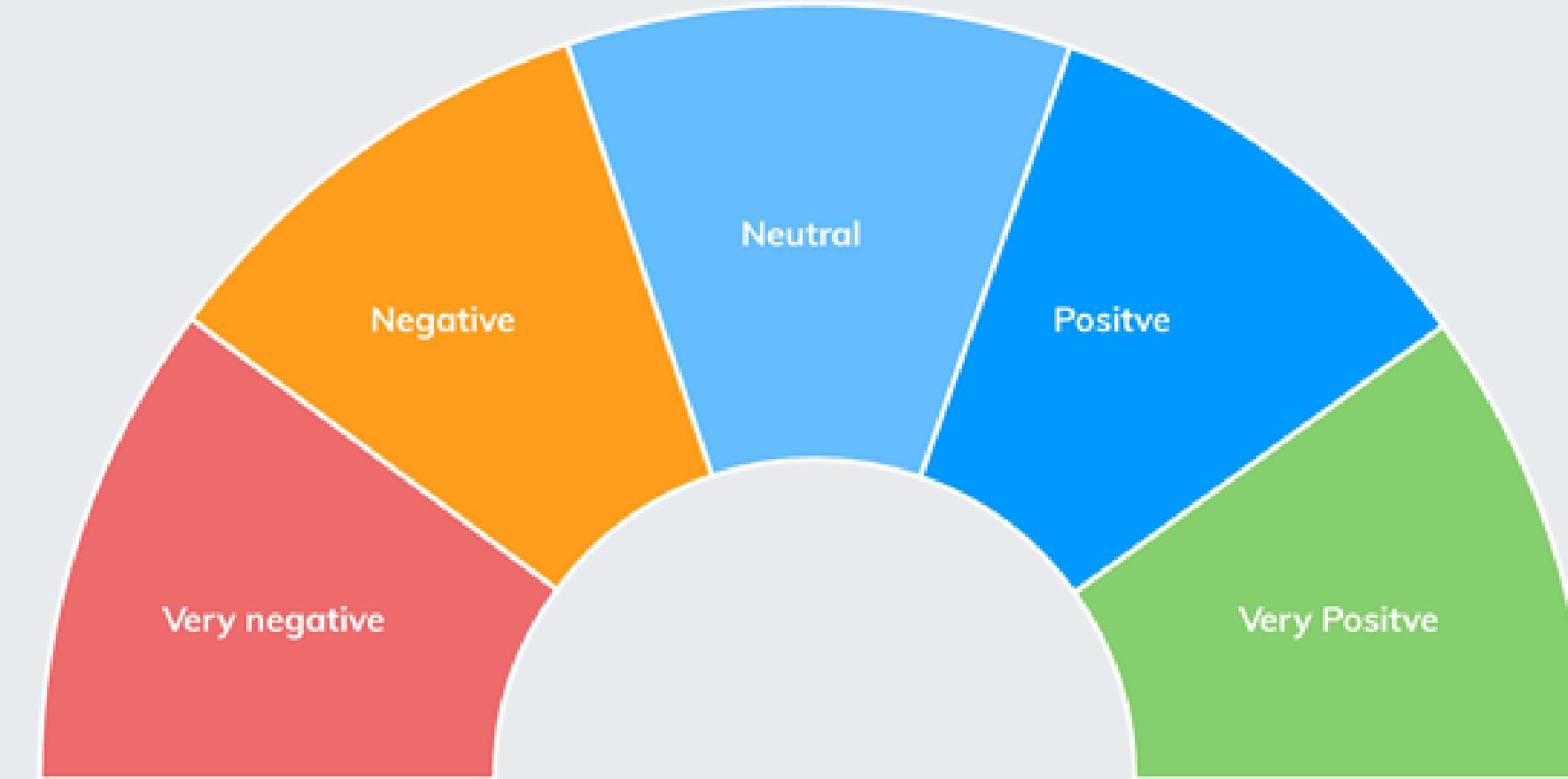
# Appliquer cette fonction aux données d'entraînement et de test
imdb_train['grouped_rating'] = imdb_train['rating'].apply(group_ratings)
imdb_test['grouped_rating'] = imdb_test['rating'].apply(group_ratings)
grouped_predictions = [group_ratings(r) for r in predictions]
```

	precision	recall	f1-score	support
1-2	0.65	0.74	0.70	7324
3-4	0.46	0.35	0.40	5176
7-8	0.45	0.36	0.40	5157
9-10	0.61	0.72	0.66	7343
accuracy			0.58	25000
macro avg	0.54	0.54	0.54	25000
weighted avg	0.56	0.58	0.56	25000

	precision	recall	f1-score	support
1-2-3-4	0.84	0.82	0.83	12500
7-8-9-10	0.83	0.85	0.84	12500
accuracy			0.84	25000
macro avg	0.84	0.84	0.84	25000
weighted avg	0.84	0.84	0.84	25000

3e et 4e Modèle IA : Attribution de grades

Attribution de Grade :
Donner un grade entre Très négatif,
Négatif, Neutre, Positif et Très
Positif



```
# Dossier contenant les fichiers .txt
train_unsup_folder = 'train/unsup'

# Lire tous les fichiers .txt du dossier
data = []
for idx, filename in enumerate(os.listdir(train_unsup_folder)):
    if filename.endswith('.txt'):
        filepath = os.path.join(train_unsup_folder, filename)
        with open(filepath, 'r', encoding='utf-8') as file:
            content = file.read()
            data.append((idx, content))

# Création de la DataFrame
df = pd.DataFrame(data, columns=['Index', 'Comment'])
```

1e solution : Dictionnaire

```
# Charger les fichiers imdb.vocab et imdbEr
vocab_file = 'imdb.vocab'
weights_file = 'imdbEr.txt'

# Lecture des fichiers
with open(vocab_file, 'r', encoding='utf-8') as vf:
    vocab = vf.read().splitlines()

weights = np.loadtxt(weights_file)

# Créer un dictionnaire associant les mots à leurs poids
word_weights = dict(zip(vocab, weights))

# Créer les listes de mots avec poids positifs et négatifs
pos_words = [word for word, weight in word_weights.items() if weight > 0]
neg_words = [word for word, weight in word_weights.items() if weight < 0]
```

```
def assign_grade(polarity):
    if polarity < -0.25:
        return 'Very Negative'
    elif -0.25 <= polarity < -0.05:
        return 'Negative'
    elif -0.05 <= polarity < 0.05:
        return 'Neutral'
    elif 0.05 <= polarity < 0.25:
        return 'Positive'
    else:
        return 'Very Positive'
```

```
# Indices des mots positifs et négatifs dans le BoW
positive_indices = [bow_vocab_index[word] for word in bow_vocab if word in positive_words]
negative_indices = [bow_vocab_index[word] for word in bow_vocab if word in negative_words]

# Calculer le score de polarité pour chaque commentaire (sans weights)
def calculate_polarity(row, positive_indices, negative_indices):
    # Somme des occurrences des mots positifs et négatifs
    pos_score = sum(row[idx] for idx in positive_indices)
    neg_score = sum(row[idx] for idx in negative_indices)

    # Calcul de la polarité en évitant la division par zéro
    if pos_score + neg_score != 0:
        return (0.8*pos_score - neg_score) / (0.8*pos_score + neg_score)
    else:
        return 0 # Si aucun mot positif ou négatif n'est trouvé

# Calcul des polarités pour chaque commentaire
print("Calcul des polarités pour chaque commentaire...")
polarities = []
# Utilisation de tqdm pour suivre la progression
for i in tqdm(range(bow_array.shape[0]), desc="Calcul des polarités", unit="commentaire"):
    row = bow_array[i] # Récupérer la ligne courante
    # Calculer la polarité manuellement (pos - neg / pos + neg)
    polarities.append(calculate_polarity(row, positive_indices, negative_indices))

# Ajouter les polarités dans le DataFrame
df['Polarity'] = polarities
```

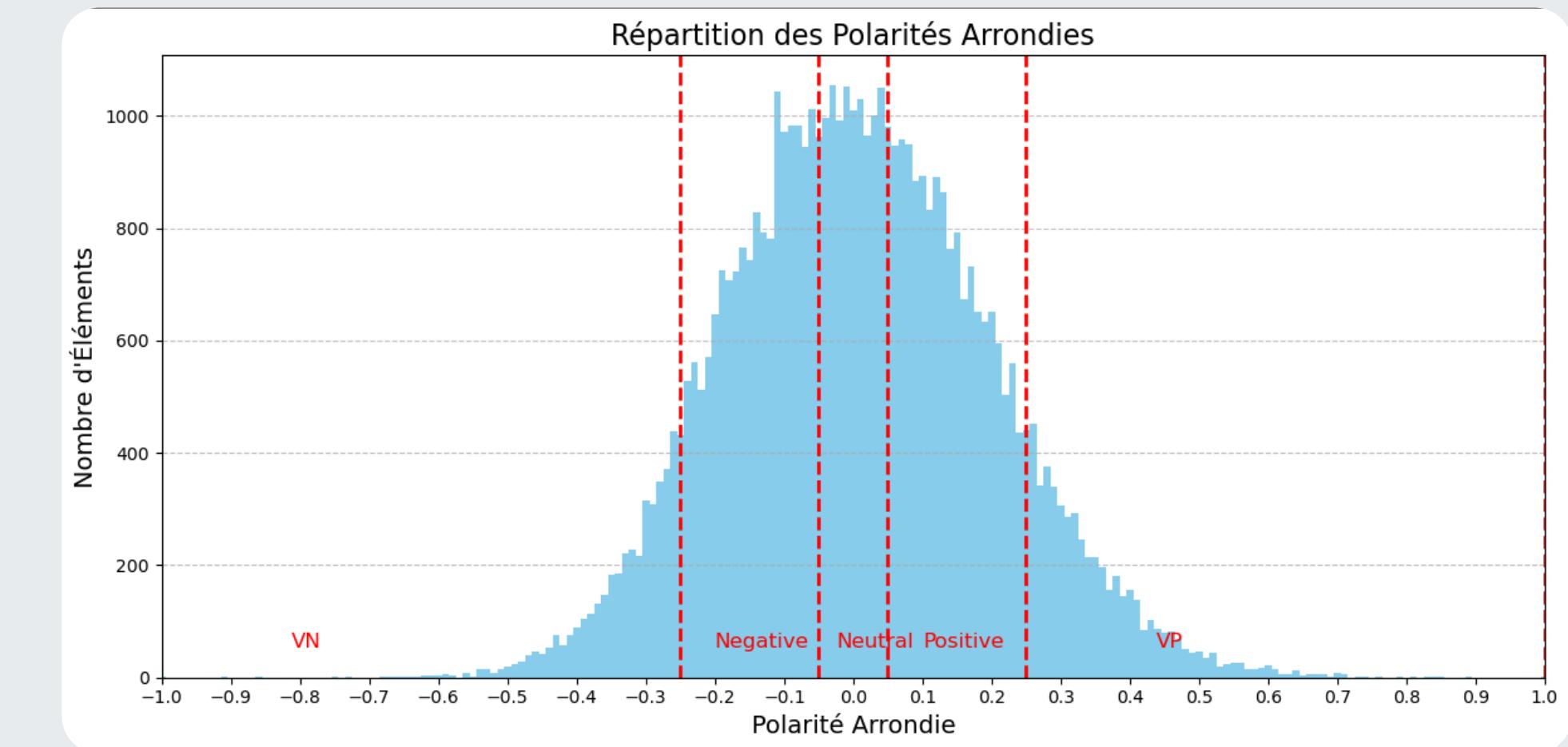
```
# Afficher les 5 premières polarités
print("Exemple de polarités calculées :")
print(df.head())
✓ 2.9s
```

Calcul des polarités pour chaque commentaire...

Calcul des polarités: 100%  50000/50000 [00:02<00:00, 17140.47commentaire/s]

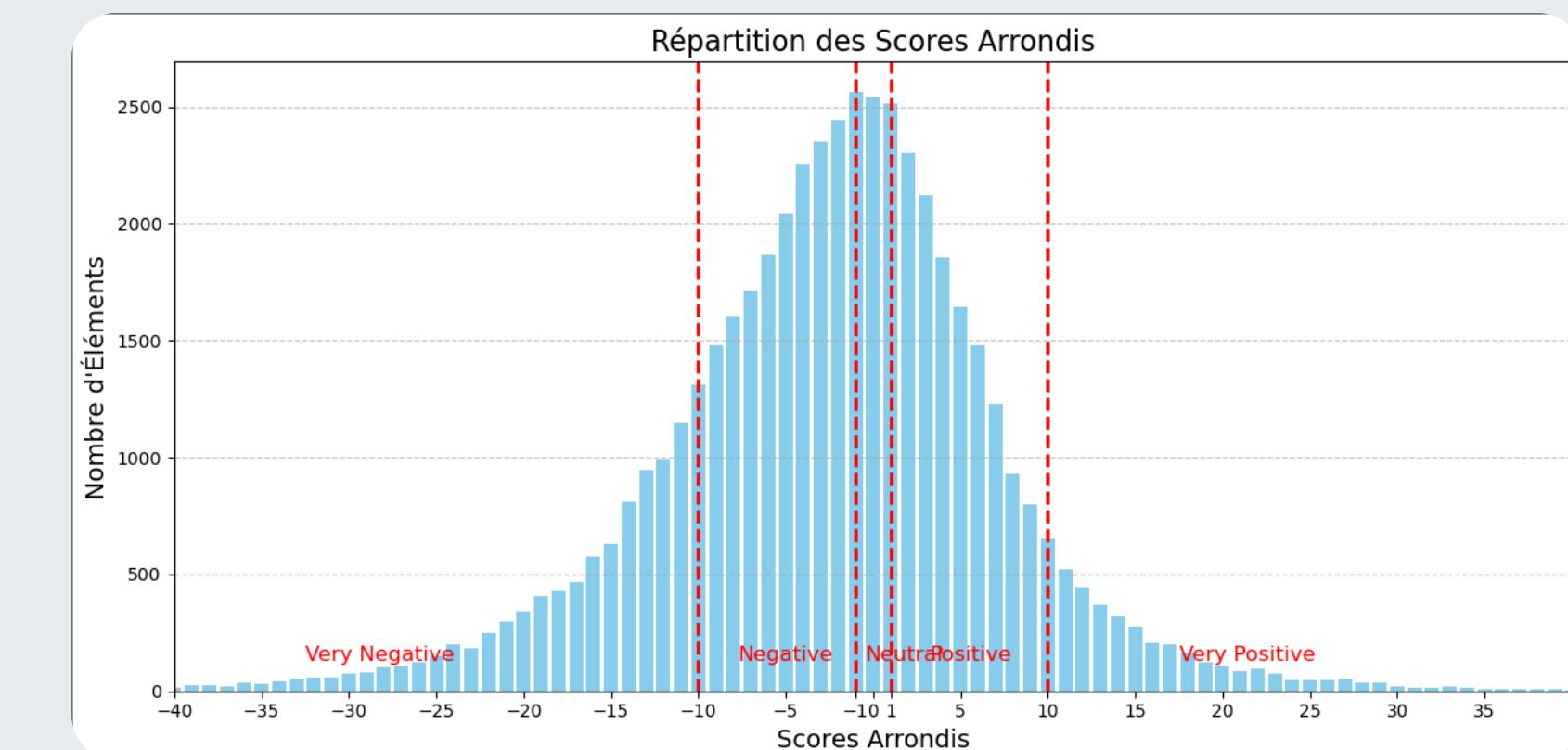
1e solution : Dictionnaire

Nombre de commentaires dans chaque catégorie :	
<i>Grade_Polarity</i>	
Negative	15621
Positive	14888
Neutral	10124
Very Positive	5277
Very Negative	4090



Alternative envisagée :

Nombre de commentaires dans chaque catégorie :	
<i>Grade</i>	
Negative	17712
Positive	13985
Very Negative	9503
Neutral	5050
Very Positive	3750



2e solution : Modèles précédents

```
print('Création du Bag of Words...')
vectorizer = CountVectorizer(binary=False, ngram_range=(1, 2), min_df=0.01)

bow_train = vectorizer.fit_transform(imdb_train['review'])
print('Bag of Words créé correctement')

print('Entraînement modèle Sentiment...')
clf_sentiment = LinearSVC(C=0.1)
clf_sentiment.fit(bow_train, train_labels)
print('Entraînement modèle Sentiment terminé...')

print('Entraînement modèle Note...')
clf_note = LinearSVC(C=0.1)
clf_note.fit(bow_train, imdb_train['rating'])
print('Entraînement modèle Note terminé...')
```

```
Dimensions de la matrice BoW : (50000, 1745)
Conversion de la matrice BoW en tableau dense...
Prédiction des notes pour chaque commentaire...
Prédiction des sentiments pour chaque commentaire...
```

```
Calcul des grades: 100% [██████████] 50000/50000 [00:04<00:00, 10817.75commentaire/s]
```

```
Nombre de commentaires dans chaque catégorie :
Grade
Very Negative    15381
Very Positive    14945
Positive          7704
Negative           6631
Neutral            5339
```

```
bow_matrix = vectorizer.transform(df['Comment'])

bow_vocab = vectorizer.get_feature_names_out()

bow_vocab_index = {word: i for i, word in enumerate(bow_vocab)}

print(f"Dimensions de la matrice BoW : {bow_matrix.shape}")

print("Conversion de la matrice BoW en tableau dense...")
bow_array = bow_matrix.toarray()

print("Prédiction des notes pour chaque commentaire...")
predicted_ratings = clf_note.predict(bow_matrix)

print("Prédiction des sentiments pour chaque commentaire...")
predicted_sentiments = clf_sentiment.predict(bow_matrix)

def assign_grade(rating, sentiment):
    if rating <= 2 and sentiment == 'negative':
        return 'Very Negative'
    elif 3 <= rating <= 4 and sentiment == 'negative':
        return 'Negative'
    elif 7 <= rating <= 8 and sentiment == 'positive':
        return 'Positive'
    elif rating > 8 and sentiment == 'positive':
        return 'Very Positive'
    else:
        return 'Neutral'

df[['Grade', 'Note', 'Sentiment']] = pd.NA

for i in tqdm(range(len(df)), desc="Calcul des grades", unit="commentaire"):
    rating = predicted_ratings[i]
    sentiment = 'positive' if predicted_sentiments[i] == 1 else 'negative'
    df.loc[i, 'Grade'] = assign_grade(rating, sentiment)
    df.loc[i, 'Sentiment'] = sentiment
category_counts = df['Grade'].value_counts()

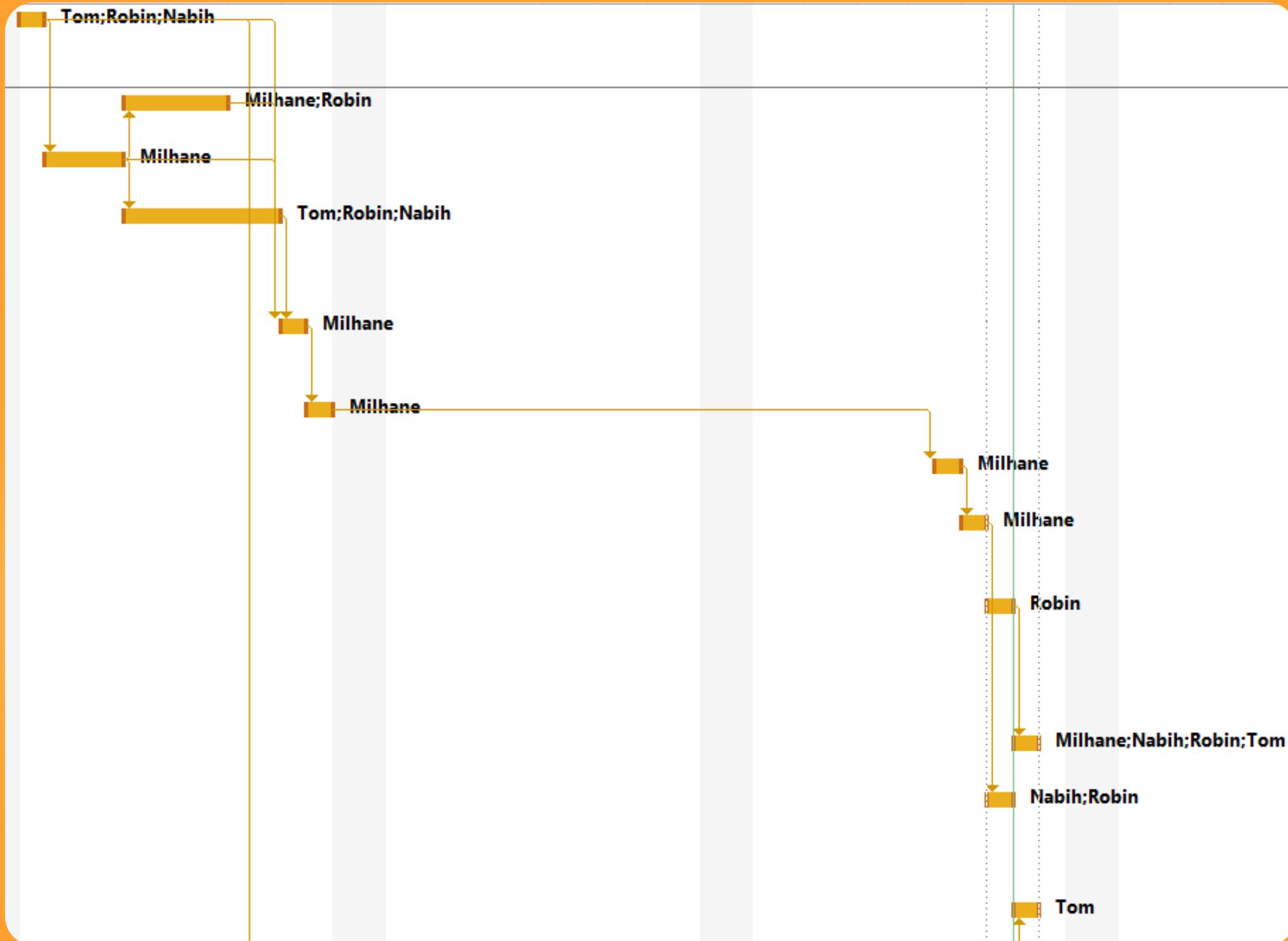
print("Nombre de commentaires dans chaque catégorie :")
print(category_counts)
```

Démonstration

Quel est votre film préféré ?

08

Gantt



SAE IMDB

Date du début du projet : 06/01/2025
Date de fin du projet : 14/02/2025

Diagramme de gantt

Membre du groupe : EL HOUB Nabih, FOEX Robin, LESELLIER Tom, RABEHI Milhane

TÂCHE	EFFECTIF	TEMPS EFFECTUÉ (H)	DÉBUT	FIN			S2 (20 heures)		S3 (28 heures)		S7 (20 heures)		
					06/11/2025	08/01/2025	10/01/2025	13/01/2025	15/01/2025	17/01/2025	10/02/2025	12/02/2025	14/02/2025
La base de données OLTP dans SQL Server		115	06/01/2025	15/01/2025									
Récupération / décompression des fichiers sources	Tom, Robin, Nabih	1	06/01/2025	06/01/2025									
Modélisation relationnelle	Milhane, Robin	16	10/01/2025	13/01/2025									
Conception MCD/MPD	Milhane	14	07/01/2025	13/01/2025									
Importation de données du fichier dans la base SQL server OLTP	Tom, Robin, Nabih	78	10/01/2025	15/01/2025									
Optimisation de la base de données OLTP	Milhane	6	10/01/2025	15/01/2025									
La base de données OLAP dans SQL Server		58	17/01/2025	12/02/2025									
Conception du MPD OLAP et sa création sur SQL server	Milhane	4	11/02/2025	11/02/2025									
Modélisation en étoile	Milhane	12	17/01/2025	12/02/2025									
Importation de données du fichier dans la base SQL server OLAP	Robin, Milhane	42	11/02/2025	12/02/2025									
Rapports PowerBI		14	12/02/2025	14/02/2025									
Importation et transformation les données extraites de la base de données SQL Server	Robin	2	12/02/2025	13/02/2025									
Création des rapports	Milhane, Robin, Tom, Nabih	12	12/02/2025	13/02/2025									
Mise en place des modèles IA avec Python		66	15/01/2025	12/02/2025									
Modèle IA dans Python pour classer automatiquement les critiques en positif ou négatif	Tom	4	13/01/2025	13/01/2025									
Modèle IA avec pour classer automatiquement la note sur 10	Tom, Milhane	38	15/01/2025	16/01/2025									
Modèle IA d'un score sous forme de grades pour l'évaluation	Tom	16	17/01/2025	10/02/2025									
Modèle IA d'association de score à une critique	Tom	4	10/02/2025	10/02/2025									
Finalisation modèle IA	Tom	4	10/02/2025	12/02/2025									
Livrables techniques		18	06/01/2025	14/02/2025									
Modèle et script optimisé de la base OLTP en expliquant les optimisations	Milhane, Nabih	2	12/02/2025	14/02/2025									
Abstract d'1/2 page présentant le sujet	Nabih	1	12/02/2025	14/02/2025									
Documentation de suivi de projet (Gantt Final)	Milhane, Nabih	4	06/01/2025	14/02/2025									
Modèle IA avec python	Tom, Nabih	3	12/02/2025	13/02/2025									
Modèle en étoile expliqué et choix d'architecture	Milhane, Robin	2	12/02/2025	14/02/2025									
Script des bases	Milhane, Robin, Nabih	1	12/02/2025	13/02/2025									
Présentation de soutenance (conclusion en anglais)	Tom	1	11/02/2025	13/02/2025									
Rapports Power BI	Milhane, Robin	4	12/02/2025	13/02/2025									

09

Conclusion

