Rapport

Bases de données
IFT 2935

par

Zakary Gaillard-Duchassin

Mohammed Aiman Rahmani

Samuel Argeris

Farley Jeannis

Mathieu Dominique Lucien Loron

présenté à

Jihene Rezgui

10 avril 2024

Université de Montréal

1 Modélisation

Nous avons été assigné le sujet 11. La tâche consistait à modéliser et implémenter une base de données qui puisse gérer l'implication des artistes dans divers films.

Le cahier des charges était le suivant:

Artiste

- Nom, prénom, âge de l'artiste.
- Adresse (et appartement au besoin).
- Renseignements personnels (habitudes, sports, relations, etc.).
- On doit également pouvoir inscrire si l'artiste a déjà joué dans des films ou théâtres (et avec quelles compagnies).
- Exigences de rémunération.

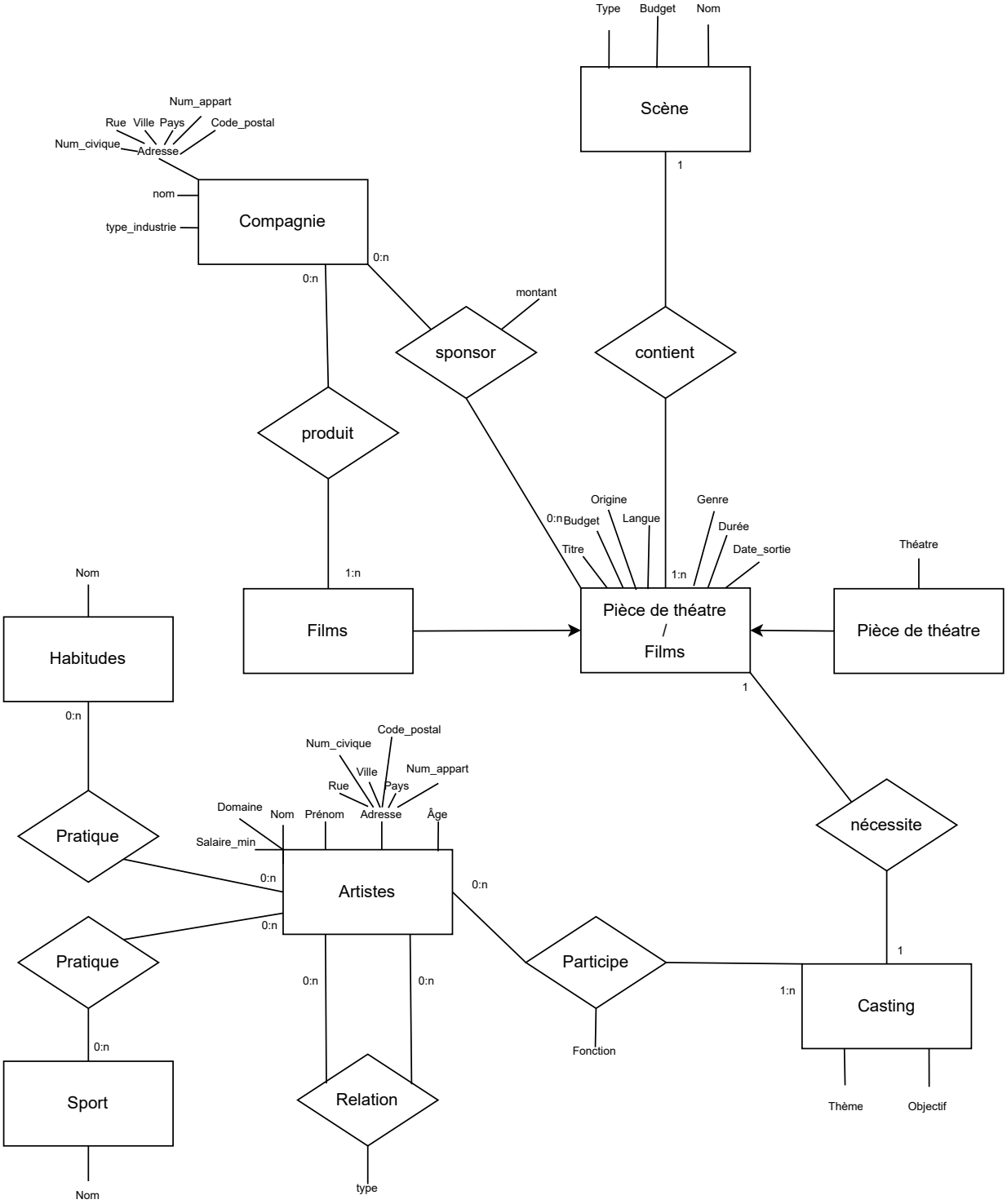
D'un autre côté, on veut aussi garder des renseignements sur le casting, l'objectif du casting et le thème. On a aussi les différentes scènes du scénario, avec les exigences en termes de réalisation de scène (action, bataille, passion, discours, etc.).

On aura aussi besoins d'enregistrer les noms des sponsors attachés aux films avec les montants des budgets, les détails de financement, etc.

1.1 Analyse des besoins

- Les artistes peuvent avoir des habitudes et des sports.
- Les artistes peuvent avoir des relations avec d'autres artistes.
- Les pieces de théâtre et les films ont des scènes.
- Les films et les pièces de théâtre ont des sponsors.
- les pièces de théâtre et les films ont des castings.
- Les castings ont des artistes qui y participent.
- Les films ont des studios de production (compagnies).

1.2 Modèle Entité-Association



1.3 Modèle Relationnel

- **Oeuvre**(id_oeuvre, titre, budget, date_sortie, durée, origine, langue, genre)
- **Films**(#id_oeuvre, #id_Studio)
- **Pièces_Théâtre**(#id_oeuvre, théâtre)
- **Scènes**(id_scène, titre, budget, type, #id_oeuvre)
- **Adresses**(id_adresse,no_civique, rue, ville, code_postal, pays, no_appartement)
- **Habitude**(id, nom)
- **Sport**(id, nom)
- **Artistes**(id_artiste, nom, prénom, date_naissance, salaire_min, domaine, #id_adresse)
- **Casting**(#id_oeuvre, objectif, thème)
- **Casting_Artiste**(#id_artiste, #id_oeuvre, fonction, salaire, date_debut, date_fin)
- **Relation**(#id_artiste1, #id_artiste2, type_relation)
- **Artiste_Sport**(#id_artiste, #id_sport)
- **Artiste_Habitude**(#id_artiste, #id_habitude)
- **Compagnies**(id_compagnie, nom, type_industrie, #id_adresse)
- **Sponsors**(#id_oeuvres #id_compagnie, montant)
- **Producteurs**(#id_oeuvres #id_compagnie)

1.4 Dépendances fonctionnelles

Oeuvre(id_oeuvre, titre, budget, date_sortie, durée, origine, langue, genre)

- id_oeuvre → titre, budget, date_sortie, durée, origine, langue, genre

Addresses(id_adresse, no_civique, rue, ville, code_postal, pays, no_appartement)

- id_adresse → no_civique, rue, ville, code_postal, pays, no_appartement

Habitude(id_habitude, nom)

- id_habitude → nom
- nom → id_habitude

Sport(id_sport, nom)

- id_sport → nom
- nom → id_sport

Compagnies(id_compagnie, nom, type_industrie, #id_adresse)

- id_compagnie → nom, type_industrie, id_adresse

Films(#id_oeuvre, #id_studio)

- id_oeuvre → id_studio

Pièce_Théâtre(#id_oeuvre, théâtre)

- id_oeuvre → théâtre

Scènes(id_scène, titre, budget, type, #id_oeuvre)

- id_scène → titre, budget, type, id_oeuvre
- id_oeuvre, titre → id_scène, type, budget

Artiste(id_artiste, nom, prénom, date_naissance, salaire_min, domaine, #id_adresse)

- id_artiste → nom, prénom, date_naissance, salaire_min, domaine, id_adresse

Casting(#id_oeuvre, objectif, thème)

- id_oeuvre → objectif, thème

Sponsor(#id_oeuvre, #id_compagnie, montant)

- id_oeuvre, id_compagnie → montant

Producteur(#id_oeuvre, #id_compagnie)

Casting_Artiste(#id_artiste, #id_oeuvre, fonction, salaire, date_début, date_fin)

- id_artiste, id_oeuvre → fonction, salaire, date_début, date_fin

Relation(#id_artiste1, #id_artiste2, type_relation)

- id_artiste1, id_artiste2 → type_relation

Artiste_Sport(#id_artiste, #id_sport)

Artiste_Habitude(#id_artiste, #id_habitude)

1.5 Normalisation

Ici il est important de noter que nous avons modélisé les relations en prenant en compte que nous avions à normaliser la base de données par la suite. Nous avons donc pris soin de décomposer les relations au maximum.

Transformation en 1NF

Ici rien à faire, car les tables sont déjà en 1NF: chaque attribut est atomique.

Transformation en 2NF

Ici rien à faire, car les tables sont déjà en 2NF: chaque attribut non-clé ne dépend pas d'une partie de la clé.

Transformation en 3NF

Ici rien à faire, car les tables sont déjà en 3NF: tout attribut n'appartenant pas à la clé ne dépend pas d'un attribut non clé

2 SQL

Tout les fichiers sql sont dans le dossier database.

2.1 LDD

Le fichier CreateUpdated.sql contient la création de la base de données et des tables.

2.2 LMD

Le fichier populate.sql contient le peuplement de la base de données. Ce fichier utilise des procédures stockées pour generer certaines données aléatoires. Les procédures stockées sont définies dans les fichiers GenCastingArtistes.sql, GenArtisteSport.sql et GenArtisteHabit.sql

2.3 Requêtes

Nous avons d'abord créer dix requêtes pour tester la base de donnée. Ces requêtes sont dans le fichier request.sql.

Par la suite, nous avons créé une classe python pour exécuter des requêtes SQL. Cette classe est dans le fichier DBManager.py. Cette classe utilise `pymssql` pour se connecter à la base de données. Elle permet d'exécuter des requêtes simples et de récupérer les résultats. Elle permet aussi d'exécuter des procédures stockées.

Durant le développement de l'application, nous avons trouvé qu'il était plus judicieux de créer des procédures stockées pour les requêtes qui nécessitaient des requêtes plus complexes. Ces procédures stockées sont dans les fichiers :

- DefAddAdresse.sql
- DefAddArtist.sql
- DefAddCasting.sql
- DefAddMovies.sql
- DefAddPlays.sql
- DefGetArtistHabit.sql
- DefGetArtistRelations.sql
- DefGetArtistSports.sql
- DefGetArtists.sql
- DefGetCastingArtists.sql
- DefGetCastings.sql
- DefGetCompagnies.sql
- DefGetMovies.sql
- DefGetPlays.sql

3 Application

Nous avons choisi de développer l'application en python. Nous avons utilisé la librairie `tkinter` ainsi que `ttkbootstrap` pour un design plus moderne des widgets.

La base de données est en SQL Server. Nous avons utilisé la librairie `pymssql` pour se connecter à la base de données.

L'application permet de visualiser les artistes, leur relations leurs habitudes et sports. Elle permet aussi de visualiser les castings et les artistes qui y participent, ainsi que les films et les pièces de théâtre présents dans la base de données.

Une fonctionnalité de recherche est aussi disponible lorsqu'on visualise les données.

Nous avons aussi ajouté une fonctionnalité pour ajouter des données dans la base de données directement depuis l'application.

Voici quelques captures d'écran de l'application:

3.1 Page d'accueil

3.2 Menu de navigation

Nous avons choisi d'ajouté un "menubar" pour naviguer entre les différentes pages de l'application à partir de n'importe quelle page.

3.3 Menu de recherche

3.4 Page de visualisation des artistes

3.5 Page de visualisation des castings

3.6 Page de visualisation des films

3.7 Page d'ajout de données

3.8 Page d'ajout d'artiste

3.9 Message de succès