Grenoble INP ENSIMAG

2020

Documentation Base de Données

Anas MEJGARI

Hatime AIT ISHAQ

Anas BENALLA

Soufiane TAGHY

Responsable projet :

Comignani Ugo

Bobineau Christophe

11/12/2020

1. Introduction

Le but du présent projet de base de données est de traiter le sujet suivant :

- Proposer une gestion de base de données pour une société multimédia
- Mettre en oeuvre nos compétences en systèmes de gestion de base de données relationnelles

Le projet sera traité de la manière suivante :

- Analyse du problème
- Conception Entité Association
- Traduction en relationnel
- Analyse des fonctionnalités
- Bilan du projet

2. Analyse du problème

Dans ce projet, on nous a demandé de concevoir et d'implémenter la base de données décrivant les fichiers multimédias hébergés sur le serveur. Tout d'abord, nous avons analysé le problème à partir de la description de l'application. Nous avons par conséquent pu déterminer un ensemble de données et de contraintes.

Les données serviront à créer les dépendances fonctionnelles entre les attributs.

Cependant, nous aurons des contraintes sur ces relations que nous devrons prendre en considération. Ensuite, nous allons modéliser ce travail d'analyse par un schéma Entité/Association, qu'on pourra traduire par la suite en un schéma relationnel.

Le schéma relationnel sera implémenté en SQL.

Une analyse à partir de la description de l'application par la méthode vu en cours a été réalisée. Nous avons détaillé cette partie analyse dans les slides de la soutenance.

- Explications à propos de quelques choix et la présentation de quelques aspects techniques du code :

- Après avoir visité plusieurs plateformes qui présentent les mêmes services que Klex, on a pu constater que certaines données ne peuvent pas être laissées à l'utilisateur final pour les entrer tels que la langue, les catégories de films et de musiques et les codecs. Ainsi, on a opté pour la même solution en implémentant un interface graphique, afin de permettre à notre client de choisir une donnée qui lui sera utile, et éviter le fait qu'il saisit certaines données de manières fausses, ce qui va nous mener à une base corrompue (par exemple l'utilisateur peut saisir 'Francais' au lieu de 'Français' et on aura donc deux représentations différentes pour la même langue alors qu'une entre ces deux est fausse). Un autre argument qui justifie notre choix, est que des fois l'utilisateur n'a pas une forte connaissance sur le domaine et peut-être bloqué lors de la saisie du codec par exemple.

- Les mêmes tests effectués pour l'ajout, la sélection et la suppression d'un film qui ont été expliqués lors de la vidéo de démonstration du code, sont applicables aussi pour les albums d'une manière identique.

- Pour la suppression des Film , on a choisi d'effectuer la suppression des fichiers liés à ce film sans supprimer les informations propres à ce film de notre base de données , de façon à élargir notre base de données et que ces données soient au service d'un autre utilisateur en cas de besoin .

- Pour la sélection des pistes musicales nous avons opté pour une sélection à travers les catégories de l'Album l'utilisateur choisit donc les albums correspondantes à cette catégorie , puis l'album , puis la piste musicale voulue.

- Pour les artistes on a rencontré une petite contrainte c'est que avec la manière dont on a choisi d'interagir avec l' utilisateur depuis nos service de base de données lors d'une éventuelle sélection , on ne peut pas demander à un utilisateur de choisir un idAlbum qui correspond à la clé de L'Artiste ,on ne peut qu'utiliser les autres attributs comme Nom , ou Spécialité qui ne peuvent pas définir un artiste de manière unique . Pour résoudre ce problème on a pensé à introduire d'autres attributs de l'Artiste tel que la nationalité et l'âge.

2.1. Données

L'analyse du sujet nous a permis de créer plusieurs données :

{Mail, Nom,Prenom,

AGE,LangueUtilisateur, CodeAccess,idFichier, Taille, Date_Depot,NumFlux,Codec,

Debit, NumFluxVideo,Largeur,Hauteur,NumFluxAudio,Echantillonage,LangueA,

NumFluxTexte,LangueTxt,dAlbum,Titre,IdArtiste,dateSortie,url_album,NumPiste,

Titre, Duree,Nom,URL_photo,Specialite,Marque,Model,Resume,largeurMax,

hauteurMax ...}

Nous allons donc mettre en œuvre toutes les données pour créer nos dépendances fonctionnelles avec les contraintes, qui nous permettront de concevoir par la suite le schéma Entité/Association qu'on traduira en schéma relationnel.

2.2. Dépendances fonctionnelles et contraintes

1. Dépendances fonctionnelles

Les dépendances ont toutes été normalisées : En effet, nous avons utilisé l'algorithme de décomposition et l'algorithme de synthèse vu en cours.

- Mail → Nom, Prenom, AGE, LangueUtilisateur, CodeAccess

3FN BCK

- idFichier → Taille,Date_Depot,Mail

3FN BCK

- NumFlux → Codec, Debit, idFichier

3FN BCK

- NumFluxVideo → Codec,Largeur,Hauteur

3FN BCK

- NumFluxAudio → Codec,Echantillonage,LangueA, Codec

3FN BCK

- NumFluxTexte → LangueTxt,Codec

3FN BCK

- IdAlbum → Titre,IdArtiste,dateSortie,url_album

3FN BCK

- NumPiste,IdAlbum → Titre, Duree

3FN BCK

- IdArtiste → Nom,URL_photo, Specialite

3FN BCK

- Marque, Modele → Resume,largeurMax,hauteurMax

3FN BCK

2. Contraintes de valeurs

· Âge, debit, min_age, taille, duree_piste > 0

· EXT(num_artiste_film) ∪ EXT(num_artiste_music) = EXT(num_artiste)

· EXT(num_acteur) C EXT(num_artiste_film)

· EXT(num_musicien) C EXT(num_artiste_music)

3. Contraintes de multiplicité

· Titre_film, anee_sortie - - ≫ categorie_film

· Titre_film, anee_sortie -/- >> URL_photo_film

· Id_fichier -- ≫ num_flux

· Num_Piste, IdAlbum -- ≫ Cat_Musique

· Num_artiste -/- > date_naissance

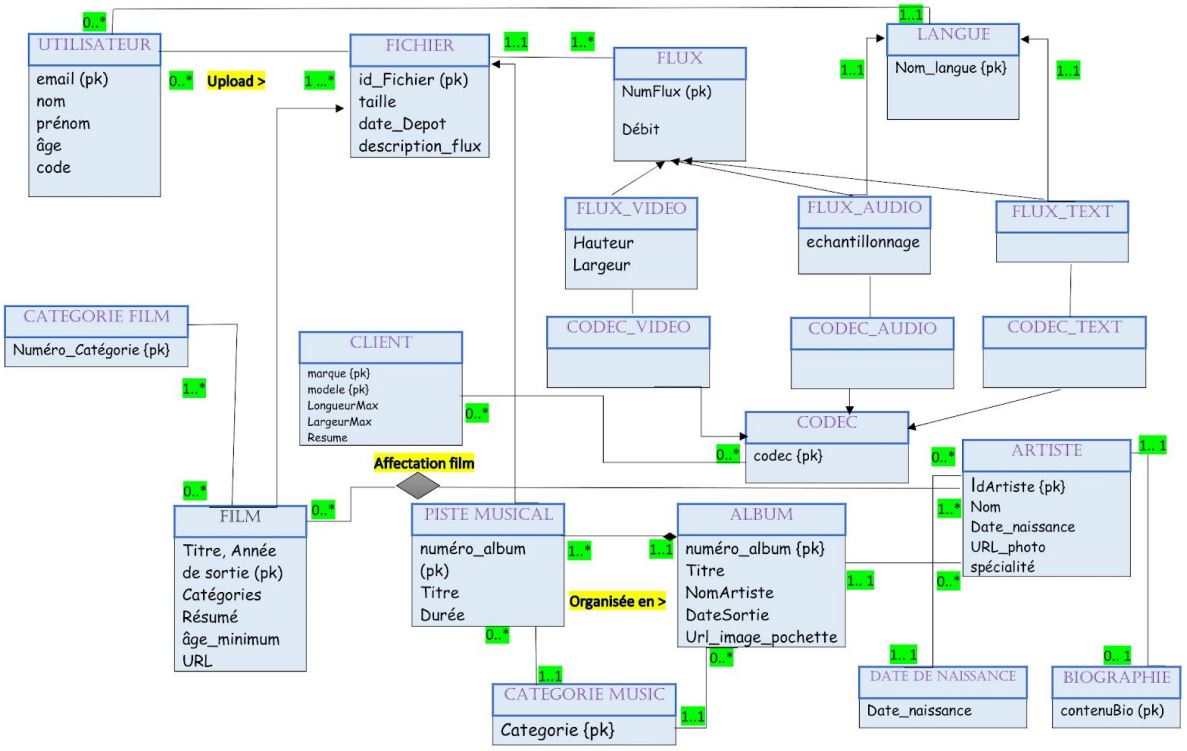
- Num_artiste -/- > CourteBiographie
- Marque_client, Model_client -- ≫ Codec_audio
- Marque_client, Model_client -- ≫ Codec_video
- Marque_client, Model_client -- ≫ Codec_text
- Num_musicien -- ≫ instrument
- Id_fichier -/- > titre_film
- Id_fichier -/- > annee_sortie
- Titre_film, annee_sortie -/- > id_fichier
- Id_fichier -/- > num_piste
- Num_piste -- ≫ id_fichier

4. Contraintes contextuelles

Un film possède un âge minimum pour pouvoir être vu.

3. Conception Entité/Associations

Nous avons construit le schéma Entité/Association grâce à l'analyse précédente :



4. Schéma relationnel

Remarque : Pour des raisons de coding style, certains noms de variables ont été légèrement modifiés par rapport à leurs noms dans la partie analyse : comme par exemple num_flux qui devient NumFlux.

Commençons par construire le schéma relationnel à partir des multiplicités indiquées dans le schéma entités-associations :

- DateDeNaissance [1..1]⟶ Artiste [0..*]

aPourDateDeNaissance(idArtiste, DateDeNaissance)

idArtiste non nul et référence Artiste

DateDeNaissance référence DateNaissance

Vérifier qu'un artiste ait exactement une date de naissance

- Film [1..1]⟶ Fichier [1..*]

FichierFilm(idFichier, Titre, anneeSortie)

idFichier non nul et référence Fichier

Titre, anneeSortie référence Film

Vérifier qu'un film ait au moins un fichier

- PisteMusicale [1..1]⟶ Fichier [1..*]

FichierPiste(idFichier, idAlbum, NumPiste)

idFichier non nul et référence Fichier

idAlbum, NumPiste référence PisteMusicale

Vérifier qu'une piste ait au moins un fichier

➢ Film [0..*]⟶ CategorieFilm [1..*]

AssocFilmCat(Categorie, Titre, anneeSortie)

Categorie référence CategorieFilm

Titre, anneeSortie référence Film

Vérifier qu'un film ait au moins une catégorie

➢ Album [0..*]⟶ CategorieMusique [1..*]

AssocAlbumCat(Categorie, idAlbum)

Categorie référence CategorieMusique

idAlbum référence Album

Vérifier qu'un album ait au moins une catégorie

➢ PisteMusicale [0..*]⟶ CategorieMusique [1..*]

AssocPisteCat(idAlbum, Categorie, NumPiste)

Categorie référence CategorieMusique

idAlbum, NumPiste référence PisteMusicale

Vérifier qu'une piste ait au moins une catégorie

➢ Film [0..*]⟶ Artiste [1..*]

aPourRole(idArtiste, Titre, anneeSortie, nomDuPersonnage, Categorie)

idArtiste référence Artiste

Titre, anneeSortie référence Film

Vérifier qu'un film ait au moins un artiste

On obtient ainsi le **schéma relationnel** suivant :

Langue(Nom_langue)

Utilisateur(Mail, Nom, Prenom, AGE, CodeAcces, LangueUtilisateur

Fichier(idFichier, Taille, Date_Depot, Mail)

Flux(NumFlux, Debit, idFichier);

Codec(Codec)

CodecAudio(Codec);

CodecTexte(Codec)

CodecVideo(Codec)

FluxVideo(NumFlux, Codec, Largeur, Hauteur);

FluxAudio(NumFlux, Echantillonage, LangueA, Codec)

FluxTexte(NumFlux, LangueTxt, Codec)

Artiste(IdArtiste, Nom, URL_photo, Specialite)

Biographie(IdArtiste, contenueBio)

DateNaissance(DateDeNaissance)

aPourDateDeNaissance(IdArtiste, DateDeNaissance)

Album(IdAlbum, Titre, IdArtiste, dateSortie, url_album)

PisteMusicale(IdAlbum, NumPiste, Titre, Duree)

Film(Titre, anneeSortie, Resume, Age, URLaffiche)

FichierFilm(Titre, anneeSortie, idFichier)

FichierPiste(idAlbum, NumPiste, idFichier)

Client(Marque, Modele, Resume, largeurMax, hauteurMax)

ClientFlux(Marque, Modele, NumFlux)

CategorieFilm(Categorie)

CategorieMusique(Categorie)

AssocFilmCat(Titre, anneeSortie, Categorie)

AssocAlbumCat(idAlbum, Categorie)

AssocPisteCat(idAlbum, NumPiste, Categorie)

aPourRole(idArtiste, Titre, anneeSortie, nomDuPersonnage, Categorie)

5. Analyse des fonctionnalités

Dans cette partie, on illustre les fonctionnalités par l'intermédiaire des transactions de manière similaire aux exemples du cours et adapté à notre base de données. On propose l'exemple suivant :

Création d'un nouvel album ou film

Begin;

Insert into Album values ("A", "titre", "idArtiste", "date", "URL")

Insert into PisteMusicale values("A", "numpiste", "titre", duree");

Delete from FichierPiste where NumPiste not in (select NumPiste from PisteMusicale

Delete from Album where idAlbum not in (select idAlbum from PisteMusicale

Commit;

Insert into Film values "titre", "annee", "resume", "age", "URL");

Commit;

Commentaire :

Pour réaliser les fonctionnalités nous avons défini les requêtes SQL2.

Nous avons proposé un exemple proche de ce qui a été vu en cours.

Les requêtes et transactions (en particulier la transaction précédente) ont été testées sur Oracle pour en vérifier leur bon fonctionnement.

Nous pouvons affirmer que les transactions fonctionnent à priori correctement.

6. Conclusion et bilan du projet

Dans ce projet, nous avons appliqué une grande partie des connaissances que nous avons apprises, ce qui peut ne pas être pleinement démontré dans le rapport. Le plus de connaissances que nous utilisons est dans la partie conception, en particulier le passage du modèle entité association au schéma relationnel.

Nous avons beaucoup appris de ce projet et nous sommes satisfait du travail qui répond aux demandes du cahier des charges. En particulier, notre base de données fonctionne comme nous le montrons dans la vidéo de démonstration de notre produit.

Enfin, notons que les habitudes d'écriture du code peuvent avoir un impact important sur l'efficacité du programme. Par conséquent, nous devons développer de bonnes habitudes d'écriture de code, supprimer le code redondant à tout moment et souvent reconstruire le code, ce qui peut grandement aider notre programmation.