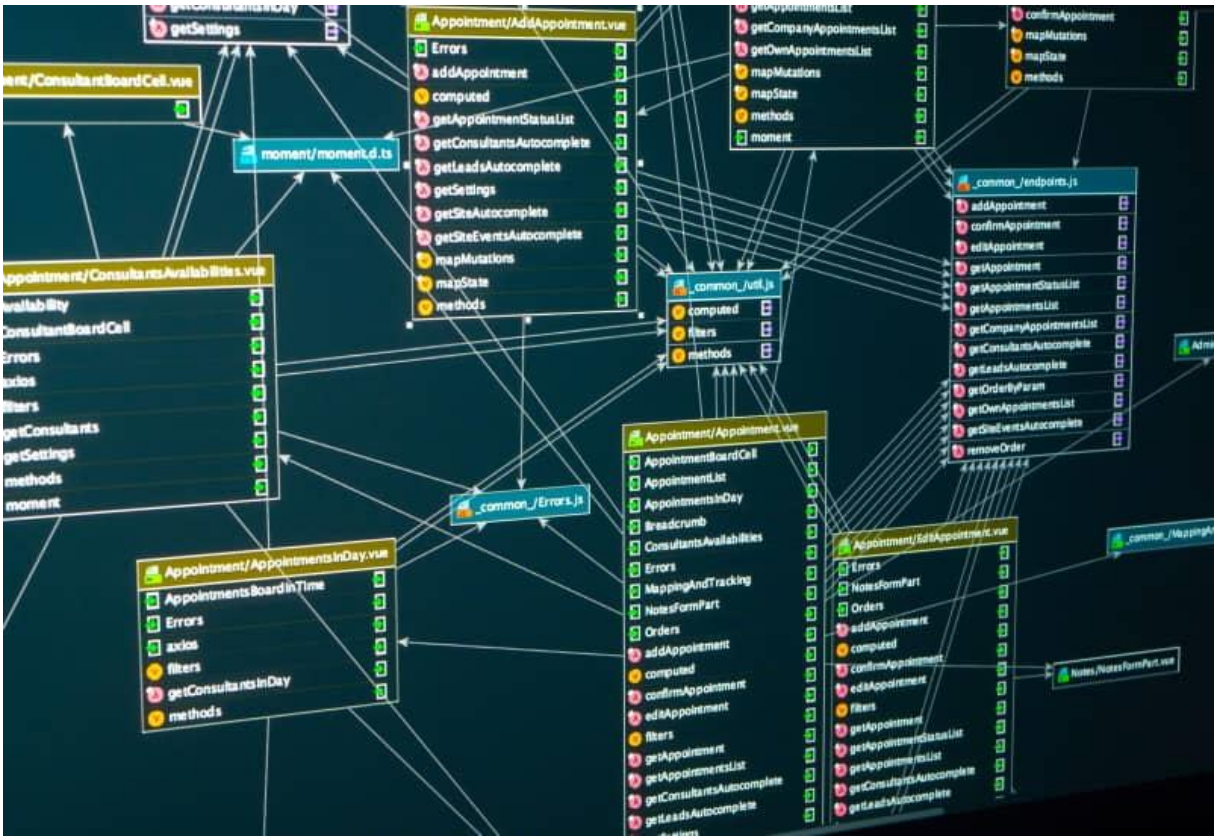
L3 INFO

LACROIX BAPTISTE
HAYE ARTHUR
BOTTIN ALEXIS
ROBERT HUGO

Université de Poitiers

BASE DE DONNEES 2
PROJET



2022/2023

I. Résumé du projet

Pour réaliser ce projet, nous nous sommes réunis en salle de TP et avons répartis les tâches équitablement. Nous avons réalisé ce projet ensemble, chacun de nous a contribué à la réalisation de ce projet, que ce soit pour la partie conception ou la partie code de la base de données.

Dans un premier temps, nous avons commencé par nous préoccuper du MCD. Nous avons réfléchi aux différentes tables pouvant être contenues dans cette base de données ainsi qu'aux différentes associations les reliant.
Nous avons réfléchi aux compositions de toutes les tables du MCD en prenant en compte les colonnes présentes dans celles-ci ainsi que les clés primaires et les clés étrangères correspondant aux liens entre chaque table du MCD.
Nous nous sommes ensuite occupés des différentes contraintes dont nous devions nous occuper par rapport à toute l'organisation que l'on a faite précédemment.
Avec cela, nous avons pu faire le script de création des tables sur Oracle, en créant toutes les tables de la base de données avec toutes les colonnes, clés primaires et étrangères et contraintes auxquelles nous avons réfléchi précédemment.
Nous avons ensuite fait le reste du travail demandés en optimisant notre base de données petit à petit afin qu'elle puisse réaliser tout ce qui était demandé.

II. Conception

Pour répondre au cahier des charges, nous l'avons décomposé et ainsi réalisé des schémas « entité-association ».

Nous avons réalisé un premier schéma. Cependant pendant la relecture du cahier des charges nous nous sommes rendu compte que ce schéma ne répondait pas entièrement aux besoins. Nous avons donc opté pour un nouveau schéma. (cf ANNEXE 1/2)

Dans le MCD sur lequel nous nous sommes basé pour développer notre base de données, nous avons tous séparés en deux parties.
En effet il y a une première partie liée au client et aux différentes façons pour qu'il accède au contenu. Le client, définit par son nom et ses coordonnées, peut soit louer directement un contenu ou en prenant un abonnement, il y a une table type définissant le type d'abonnement pris par le client contenant le nom de l'abonnement et son prix.
Il peut effectuer un achat, qui aura sa propre table étant donné que le client télécharge son achat, un achat a ainsi un format et a besoin d'un logiciel pour s'ouvrir. Etant donné qu'un achat peut être lu avec plusieurs formats différents avec des logiciels différents, on a créé une table logiciel et une table format.

La deuxième partie concerne les types de contenus et les personnes contribuant à la réalisation de ceux-ci.
Le contenu qui est acheté ou loué par le client peut être une série qui est caractérisée par son nombre de saisons, les saisons d'une série sont dans la table saison qui a un nombre d'épisodes, sa description et son évaluation.
Le contenu peut également être un film, avec la table cinéma, défini par sa durée et qui possède un ou plusieurs scénaristes et avec des acteurs qui jouent dedans.
Il y a aussi les divertissements, définis par leurs durées, ayant un ou plusieurs metteur en scènes, scénaristes et interprètes. Et enfin les programmes pour la jeunesses ayant un ou plusieurs compositeurs.
Chacune des personnes évoqués acteurs,scénaristes, metteurs en scènes, réalisateurs et interprètes sont définis par leurs noms et prénoms.

Notre projet se base donc sur ce diagramme . Certaine entités ont été modifiées aux cours du projet pour simplifier et optimiser notre base de donées.

On peut voir différents type de cardinalité entre les entités et les relations. Par exemple un réalisateur peut avoir réalisé 0 ou plusieurs contenus (0-n) mais un contenu possdède au moins 1 réalisateur. On peut voir aussi qu'un contenu est défini par un unique type(1-1) . Cependant un type peut etre associé à différents contenus(0-n).

III. TRANSACTION

L5- Specification des Transactions suivante :

T1 : création d'un compte par un client et prise d'un abonnement ;

T2 : résiliation d'un abonnement par un client (ce qui met fin à toutes ses locations en cours) ;

T3 : location d'un contenu par un abonné ;

T4 : suppression d'un contenu par un administrateur sur la plateforme (ce qui met fin à toutes les locations en cours de ce contenu).

Mais Avant de Commencer Voici une presentation des Tables Concernée/Utilisée :

TABLE **Abonner(abonnerid , abonnementid , clientid, date_deb, date_fin)**

TABLE **Abonnement(abonnementid , nom , nombreMax , prix)**

TABLE **Location_Abonner (abonnerid , contenueid, evaluation_abo)**

TABLE **Client_ABO (clientid , abonnerid)**

La Table ABONNER permet à un client de prendre un Abonnement.

Le Type d'abonnement, à savoir VIP, classique ou autre est déterminener dans la Table ABONNEMENT, ou nombreMax designe le nombre de contenue qu'il peut louer au maximum.

La Table Location_Abonner permet de connaitre a partir de quel Abonnement(ABONNER) un contenue a été louer et par consequent savoir quel contenue a été louer.On dispose aussi de l'identifiant du client via abonnerid, permettant de savoir qui loue.

La Table Client_ABO repertorie tout les Client abonnée, cette Table fait office du Statut d'abonner pour un client, mais permet également de connaitre tout les abonnement pris par un Client.

TRANSACTION T1 :

```
DECLARE
   new_clientid INTEGER;
   new_abonnerid INTEGER;
BEGIN
   SAVEPOINT create_account;
   -- Insére les informations du client dans la table Client
   INSERT INTO Client (nom, prenom, Num_rue, RUE, Ville, mail, tel, mdp)
   VALUES ('[nom du client]', '[prénom du client]', [numéro de rue], '[nom de la rue]', '[nom de la ville]', '[adresse e-mail]', [numéro de téléphone], '[mot de passe]');
   -- Récupére l'identifiant du client nouvellement créé
   SELECT MAX(clientid) INTO new_clientid FROM Client;
   -- Insére une nouvelle ligne dans la table Abonner pour cet abonné
   INSERT INTO Abonner (abonnementid, clientid, date_deb, date_fin)
   -- On va considerer ici que la prise d'un abonnement est pour une periode de 1 ans
   VALUES ([identifiant de l'abonnement], new_clientid, SYSDATE, ADD_MONTHS(SYSDATE, 12));
   -- Récupére l'identifiant de l'abonnement nouvellement créé
   SELECT MAX(abonnerid) INTO new_abonnerid FROM Abonner;
   -- Insére une nouvelle ligne dans la table Client_ABO pour associer le client à son abonnement
   INSERT INTO Client_ABO (clientid, abonnerid)
   VALUES (new_clientid, new_abonnerid);
   -- Si toutes les opérations ont été réalisées avec succès, valide la transaction
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Compte créé avec succès');
EXCEPTION
```

WHEN OTHERS THEN

-- En cas d'erreur, annuler la transaction et affiche un message d'erreur

ROLLBACK TO create_account;

DBMS_OUTPUT.PUT_LINE('Erreur lors de la création du compte : ' || SQLERRM);

END;

Ici si toutes les opérations sont réalisées avec succès, la transaction est validée. Si une erreur se produit, la transaction est annulée et un message d'erreur est affiché.

TRANSACTION T2 :

DECLARE

abonnerid **INTEGER**;

BEGIN

-- Récupére l'identifiant de l'abonnement à résilier pour ce client

SELECT abonnerid **INTO** abonnerid **FROM** Abonner **WHERE** clientid = [identifiant du client] **AND** date_fin > SYSDATE;

ICI nous n'avons pas encore tout a fait pris en compte qu'un Client ne puisse avoir qu'un seul Abonnement actif, et pour cela, alors nous utilisons la clause AND date_fin > SYSDATE qui sert à filtrer les résultats pour ne retourner que l'abonnement actif du client.

-- Supprime toutes les locations effectué via/associées à cet abonnement

DELETE FROM Location_Abonner **WHERE** abonnerid = abonnerid;

-- Supprime la ligne correspondante dans la table Client_ABO, le client perd son statut d'abonné

DELETE FROM Client_ABO **WHERE** clientid = [identifiant du client] **AND** abonnerid = abonnerid;

-- Met à jour la date de fin de l'abonnement pour refléter la résiliation

UPDATE Abonner **SET** date_fin = SYSDATE **WHERE** abonnerid = abonnerid;

-- Afficher un message de confirmation

DBMS_OUTPUT.PUT_LINE('Abonnement résilié avec succès');

EXCEPTION

WHEN NO_DATA_FOUND THEN

-- Si aucun abonnement actif n'est trouvé pour ce client a partir de la table Client_ABO, affiche un message d'erreur

DBMS_OUTPUT.PUT_LINE('Aucun abonnement actif trouvé pour ce client');

WHEN OTHERS THEN

-- En cas d'erreur, afficher un message d'erreur

```
    DBMS_OUTPUT.PUT_LINE('Erreur lors de la résiliation de l''abonnement : ' || SQLERRM);
END;
```

TRANSACTION T3 :

```
DECLARE
   p_abonnerid Abonner.abonnerid%TYPE;
   p_contenueid Contenu.contenuid%TYPE := [identifiant du contenu];
   p_evaluation_abo Location_Abonner.evaluation_abo%TYPE := [note du client sur le contenu];
        -- Le Client évalue le contenue une fois Louer
   v_nombre_locations INTEGER;
   v_nb_max_locations INTEGER;
        -- v_nb_max_locations correspond au nombre maximum de location autorisé par son abonnement
BEGIN
   -- Vérifie que l'abonné a un abonnement actif
   SELECT abonnerid INTO p_abonnerid
   FROM Abonner
   WHERE clientid = [identifiant du client] AND date_fin > SYSDATE;
 -- Calcule le nombre de locations déjà faites par le Client Abonné
   SELECT COUNT(*) INTO v_nombre_locations FROM Location_Abonner WHERE abonnerid = p_abonnerid;
   -- Vérifie que le nombre maximal de locations n'est pas dépassé
  SELECT nombreMax INTO v_nb_max_locations FROM Abonnement WHERE abonnementid = (SELECT abonnementid FROM Abonner WHERE abonnerid = p_abonnerid);
IF v_nombre_locations >= v_nb_max_locations
THEN RAISE_APPLICATION_ERROR(-20001, 'Nombre maximum de locations atteint pour cet abonné.');
   END IF;
   -- Insertion de la location
   INSERT INTO Location_Abonner (abonnerid, contenueid, evaluation_abo)
   VALUES (p_abonnerid, p_contenueid, p_evaluation_abo);
```

```
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Location enregistrée avec succès');
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Erreur lors de la location : ' || SQLERRM);
END;
```

TRANSACTION T4 :

```
DECLARE
  p_contenueid Contenue.contenueid%TYPE := [id du contenu à supprimer];
BEGIN
  -- Supprime toutes les locations de ce contenu dans la table Contenue_Louer
  DELETE FROM Contenue_Louer WHERE contenueid = p_contenueid;
  -- Supprime toutes les locations de ce contenu dans la table Location_Abonner
  DELETE FROM Location_Abonner WHERE contenueid = p_contenueid;
  -- Supprimer le contenu lui-même dans la table Contenue
  DELETE FROM Contenue WHERE contenueid = p_contenueid;
END;
/
```

L6- Spécification des verrous :

Pour la transaction de création d'un compte client T1 : il n'y a pas besoin de verrous car il s'agit d'une opération simple qui n'implique pas d'accès concurrentiel aux données.

Pour la transaction de résiliation d'un abonnement par un client T2 : il faudrait mettre un verrou sur la table Location_Abonner pour éviter qu'une autre transaction ne modifie les données de cette table pendant la suppression des locations du client. Il faudrait également mettre un verrou sur la table Client_ABO pour éviter qu'une autre transaction ne modifie les données de cette table pendant la suppression de la ligne correspondant au statut d'abonné du client.

Pour la transaction de location d'un contenu par un abonné T3 : il faudrait mettre un verrou sur la table Location_Abonner pour éviter que deux transactions différentes ne modifient les données de

cette table en même temps (par exemple pour ajouter une location). Il faudrait également mettre un verrou sur la table Abonnement pour vérifier que le nombre de locations maximum autorisé n'est pas dépassé.

Pour la transaction de suppression d'un contenu <u>T4</u> : il faudrait mettre un verrou sur la table Location_Abonner et la table Contenue_Louer sur pour éviter qu'une autre transaction ne modifie les données de cette table pendant la suppression des locations associées au contenu.

L7- Spécification du niveau d'Isolation choisi :

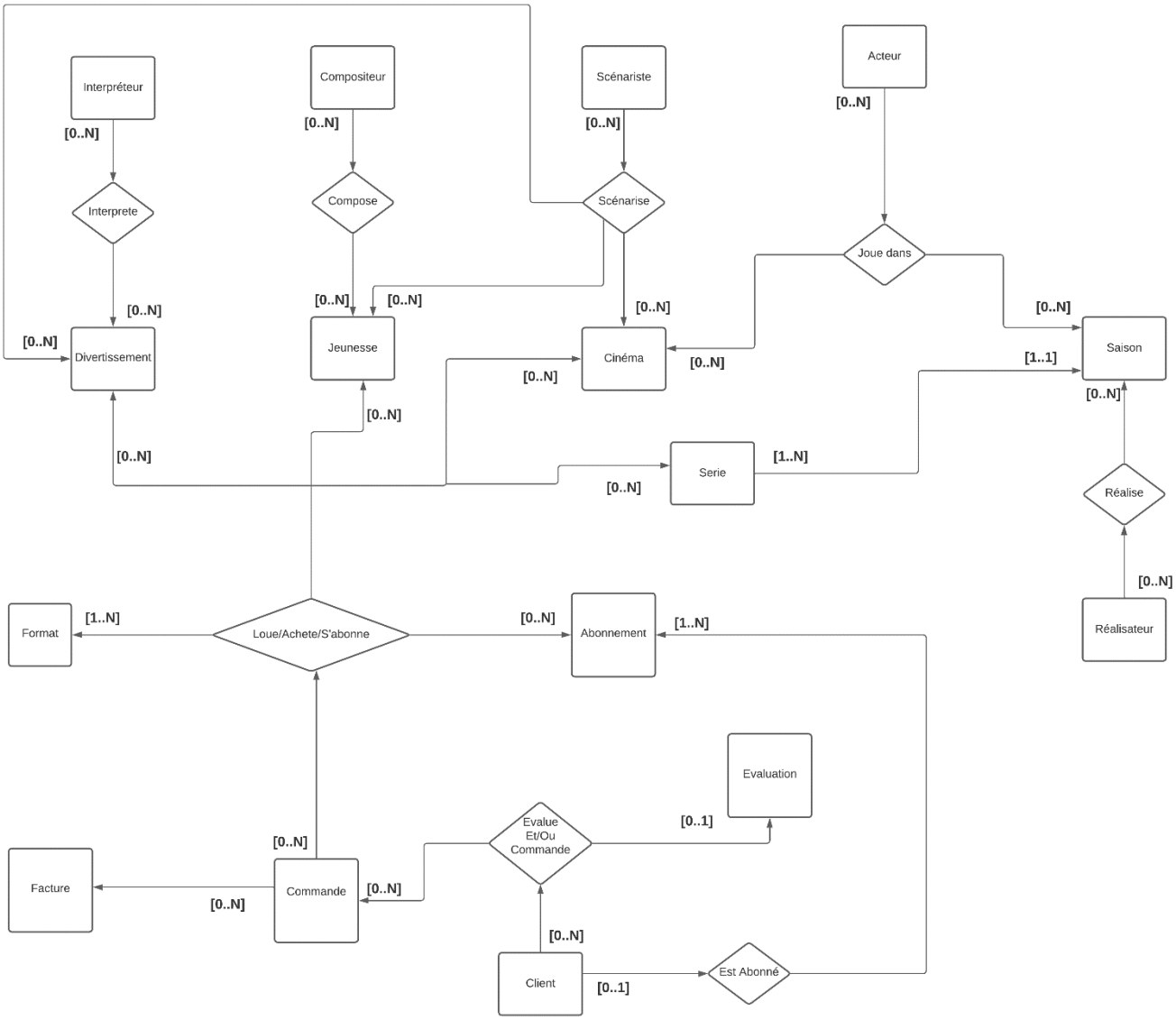
Pour cela 2 choix possible : le niveau de lecture répétable (repeatable read) ou le niveau de lecture confirmée (serializable).

- Le niveau d'isolation Repeatable Read garantit qu'une transaction ne verra pas les modifications apportées par une autre transaction en cours. Cela empêche les anomalies de concurrence telles que la lecture sale, la lecture non répétable et la lecture fantôme.
- Le niveau de lecture confirmée garantit que les transactions sont exécutées comme si elles avaient été exécutées séquentiellement, ce qui élimine les problèmes de concurrence. Cependant, cela peut entraîner des performances plus faibles en raison du blocage des verrous pendant de longues périodes.

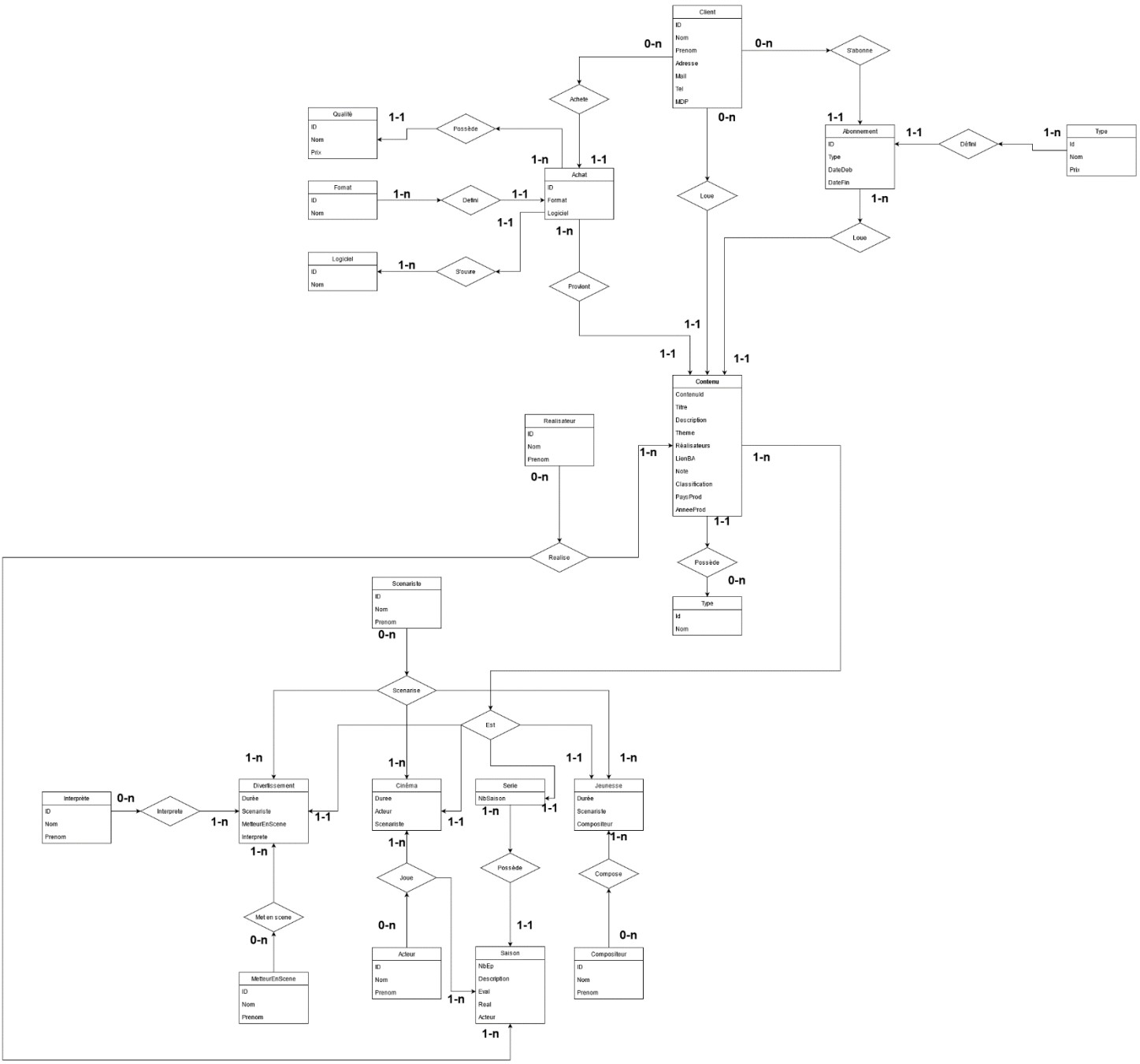
Nous préferons donc choisir le niveau de lecture répétable (repeatable read)

Annexe 1

Premier schéma



Annexe 2


Client
ID
Nom
Prenom
Adresse
Mail
Tel
MDP
S'abonne
Achete
Qualité
ID
Nom
Prix
Possède
Achat
ID
Format
Logiciel
Format
ID
Nom
Defini
Logiciel
ID
Nom
S'ouvre
Provient
Loue
Abonnement
ID
Type
DateDob
DateFin
Défini
Type
Id
Nom
Prix
Loue
Contenu
ContenuId
Titre
Description
Theme
Réalisateurs
LienBA
Note
Classification
PaysProd
AnneeProd
Realisateur
ID
Nom
Prenom
Realise
Possède
Type
Id
Nom
Scenariste
ID
Nom
Prenom
Scenarise
Est
Interprète
ID
Nom
Prenom
Interprete
Divertissement
Durée
Scenariste
MetteurEnScene
Interprete
Cinéma
Duree
Acteur
Scenariste
Serie
NbSaison
Jeunesse
Durée
Scenariste
Compositeur
Met en scene
MetteurEnScene
ID
Nom
Prenom
Joue
Acteur
ID
Nom
Prenom
Possède
Saison
NbEp
Description
Eval
Real
Acteur
Compose
Compositeur
ID
Nom
Prenom
0-n
1-1
1-n