# Rapport Laboratoire no 2

Sahitaj Edison, Nicolet Victor et Pilet Théo

BDR

Novembre 2024

# Table des matières

# Introduction

Ce rapport détaille les travaux réalisés dans le cadre du laboratoire, visant à modéliser et manipuler une base de données. Il couvre la transformation du schéma EA en modèle relationnel, la création des scripts SQL, ainsi que l'adaptation de la base pour répondre aux exigences spécifiques.

# Partie 1

La première partie consistait à transformer le schéma EA initial en un modèle relationnel, en respectant les conventions de nommage établies. Ce modèle inclut les relations principales pour les clients, les articles, les consoles, et les éditeurs, ainsi que les associations nécessaires pour gérer les achats, les mots-clés, les genres, et les contenus téléchargeables. Les contraintes d'intégrité référentielle et de cardinalité ont été définies pour assurer la cohérence des données. Voici un aperçu de notre schéma relationnel :

Client(pseudo, dateNaissance, adresseFacturation, email)
Article(id, nom, description, dateSortie, prix, note)
Article_MotCles(articleId, motsCles)
*Article_MotsCles.articleId référence Article.id*
Client_Article(pseudo, id, dateAchat)
*Client_Article.pseudo référence Client.pseudo*
*Client_Article.id référence Article.id*

Console(nom, nomFabricant anneeParution)
Article_Console(id, nom)
*Article_Console.id référence Article.id*
*Article_Console.nomFabricant référence Console.nom NOT NULL*

Fabricant(nom)
*Console.nom référence Fabricant.nom NOT NULL*

DLC(idArticle, idJeuVideo, necessiteJeuDeBase)
*DLC.idArticle référence Article.id*
JeuVideo(idArticle, idEditeur ageMinimum)
*JeuVideo.idArticle référence Article.id*
*DLC.idJeuVideo référence JeuVideo.idArticle*

Genre(nom)
JeuVideo_Genre(idJeuVideo, nomGenre)
*JeuVideo_Genre.idJeuVideo référence JeuVideo.idArticle*
*JeuVideo_Genre.nomGenre référence Genre.nom*

Editeur(id, nom, siegeSocial)
*JeuVideo.idEditeur référence Editeur.id*

## Choix des clés étrangères

Relation Client_Article :
Nous avons suivi les règles Associations binaires N:M.
Les entités Client et Article se transforment comme des types d'entité forts. Il faut donc une troisième relation qui a pour clé primaire la combinaison des clés primaires de Article et de Client.

Relation JeuxVideo_Genre :
Nous avons suivi les règles Associations binaires N:M.
Les entités JeuxVideo et Genre se transforment comme des types d'entité forts. Il faut donc une troisième relation qui a pour clé primaire la combinaison des clés primaires de JeuxVideo et de Genre. Cependant, celle de jeuxVideo est celle de son parent.

Relation Article_Console
Nous avons suivi les règles Associations binaires N:M.
Les entités Article et Console se transforment comme des types d'entité forts. Il faut donc une troisième relation qui a pour clé primaire la combinaison des clés primaires de Article et de Console

Relation entre Fabricant et Console :
Nous avons suivi les règles Associations binaires 1:N.
La clé étrangère qui référence Fabricant est contenu dans Console afin d'évité la redondance dans Fabricant.

Relation entre Editeur et JeuVideo
Nous avons suivi les règles Associations binaires 1:N.
La clé étrangère qui référence Editeur est contenu dans JeuVideo afin d'évité la redondance dans Editeur.

Relation entre DLC et JeuVideo
Nous avons suivi les règles Associations binaires 1:N.
En plus de la clé étrangère qui référence le parent de DLC afin de ne pas sur ajouter de contrainte par la suit, la clé étrangère qui référence JeuVideo est contenu dans DLC afin d'évité la redondance dans JeuVideo.

Relation dans Article
Nous avons suivi les règles Attributs multivalués.
L'attribut motCles devient une relation indépendante, avec une clé étrangère qui référence la table Article.

# Partie 2

Un script SQL a été implémenté pour créer la structure de notre base de données tout en respectant le schéma relationnel. De plus nous avons implémenté un maximum de contraintes d'intégrité.

## Intégrité référentielle

- SET NULL sur console

L'association entre « console » et « fabricant » indique que chaque console a un fabricant, mais si un fabricant est supprimé, cela ne signifie pas forcément qu'on doit supprimer les consoles qui sont associées

En laissant le champs « nomFabricant » à NULL via SET NULL, cela permet de préserver les consoles car elles pourraient être réassignées à un autre fabricant.

- Autre choix CASCADE

Les relations comme « Client_Article » ou « Article_Console » impliquent une forte dépendance. Si un article est supprimé, il est logique que les enregistrements disparaissent également.

## Contraintes de validation

- Si Article.dateSortie n'est pas NULL alors le prix non plus

Dans la table Article :

```
CONSTRAINT CK_Article_dateSortie CHECK (dateSortie IS NULL OR prix IS NOT NULL),
```

- Article.note peut être non NULL que si on est après sa dateSortie

Dans la table Article :

```
CONSTRAINT CK_Article_note CHECK (note IS NULL OR dateSortie <= CURRENT_DATE)
```

- Un article doit avoir une datesSortie NULL ou >= à l'annéeParution de toutes les consoles sur lesquelles il se joue

Impossible à implémenter la contrainte car cela demande une vérification sur plusieurs relations et ne peut pas être exprimé avec un CHECK

- Un client ne doit pas pouvoir acheter un Article s'il n'a pas l'âge minimum

Impossible à implémenter car cela demande une comparaison sur plusieurs relations et ne peut pas être exprimé avec un CHECK

- Un client ne peut pas avoir un pseudo de plus de 80 caractères

Dans la table client :

```
pseudo VARCHAR(80),
```

## Contrainte d'unicité

Les champs « email » dans la table client et « nom » dans la table éditeur ont besoin d'une contrainte d'unicité

```
CONSTRAINT UC_Client_email UNIQUE (email)
```

```
CONSTRAINT UC_Editeur_nom UNIQUE (nom)
```

## Choix personnel

Nous avons décidé de fixer le nombre de chiffre après la virgule sur le champ « prix » de la table Article à 2

```
prix DECIMAL(10,2),
```

Script SQL

```
DROP TABLE IF EXISTS Client;
DROP TABLE IF EXISTS Article;
DROP TABLE IF EXISTS Article_MotsCles;
DROP TABLE IF EXISTS Client_Article;
DROP TABLE IF EXISTS Console;
DROP TABLE IF EXISTS Article_Console;
DROP TABLE IF EXISTS Fabricant;
DROP TABLE IF EXISTS DLC;
DROP TABLE IF EXISTS JeuVideo;
DROP TABLE IF EXISTS JeuVideo_Genre;
DROP TABLE IF EXISTS Genre;
DROP TABLE IF EXISTS Editeur;

CREATE TABLE Client (
   pseudo VARCHAR(80),
   dateNaissance DATE NOT NULL,
   adresseFacturation VARCHAR(255) NOT NULL,
   email VARCHAR(255),
   CONSTRAINT PK_Client PRIMARY KEY (pseudo),
   CONSTRAINT UC_Client_email UNIQUE (email)
);

CREATE TABLE Article (
   id SERIAL,
   nom VARCHAR(255) NOT NULL,
   description VARCHAR(255) NOT NULL,
   dateSortie DATE,
   prix DECIMAL(10,2),
   note INTEGER,
   CONSTRAINT PK_Article PRIMARY KEY (id),
   CONSTRAINT CK_Article_dateSortie CHECK (dateSortie IS NULL OR prix IS
NOT NULL),
   CONSTRAINT CK_Article_note CHECK (note IS NULL OR dateSortie <=
CURRENT_DATE)
);

CREATE TABLE Article_MotsCles (
   articleId SERIAL,
   motCle VARCHAR(255),
   CONSTRAINT PK_Article_MotsCles PRIMARY KEY (articleId, motCle),
   CONSTRAINT FK_Article_MotsCles_articleId FOREIGN KEY (articleId)
REFERENCES Article(id)
   ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Client_Article (
   pseudoClient VARCHAR(80),
   idArticle SERIAL,
   dateAchat DATE NOT NULL,
```

```
   CONSTRAINT PK_Client_Article PRIMARY KEY (pseudoClient, idArticle),
   CONSTRAINT FK_Client_Article_pseudoClient FOREIGN KEY (pseudoClient)
REFERENCES Client(pseudo)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT FK_Client_Article_idArticle FOREIGN KEY (idArticle)
REFERENCES Article(id)
      ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Fabricant (
   nom VARCHAR(255),
   CONSTRAINT PK_Fabricant PRIMARY KEY (nom)
);

CREATE TABLE Console (
   nom VARCHAR(255) NOT NULL,
   anneeParution INTEGER NOT NULL,
   nomFabricant VARCHAR(255) NOT NULL,
   CONSTRAINT PK_Console PRIMARY KEY (nom),
   CONSTRAINT FK_Console_nomFabricant FOREIGN KEY (nomFabricant)
REFERENCES Fabricant(nom)
      ON DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE Article_Console (
   idArticle SERIAL,
   nomConsole VARCHAR(255) NOT NULL,
   CONSTRAINT PK_Article_Console PRIMARY KEY (idArticle, nomConsole),
   CONSTRAINT FK_Article_Console_idArticle FOREIGN KEY (idArticle)
REFERENCES Article(id)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT FK_Article_Console_nomConsole FOREIGN KEY (nomConsole)
REFERENCES Console(nom)
      ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Editeur (
   id SERIAL,
   nom VARCHAR(255),
   siegeSocial VARCHAR(255) NOT NULL,
   CONSTRAINT PK_Editeur PRIMARY KEY (id),
   CONSTRAINT UC_Editeur_nom UNIQUE (nom)
);

CREATE TABLE JeuVideo (
   idArticle SERIAL,
   idEditeur SERIAL,
   ageMinimum INTEGER NOT NULL,
   CONSTRAINT PK_JeuVideo PRIMARY KEY (idArticle),
```

```
    CONSTRAINT FK_JeuVideo_idArticle FOREIGN KEY (idArticle) REFERENCES
Article(id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_JeuVideo_idEditeur FOREIGN KEY (idEditeur) REFERENCES
Editeur(id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE DLC (
    idArticle SERIAL,
    idJeuVideo SERIAL,
    necessiteJeuDeBase BOOLEAN NOT NULL,
    CONSTRAINT PK_DLC PRIMARY KEY (idArticle),
    CONSTRAINT FK_DLC_idArticle FOREIGN KEY (idArticle) REFERENCES
Article(id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_DLC_idArticleJeuVideo FOREIGN KEY (idJeuVideo)
REFERENCES JeuVideo(idArticle)
        ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Genre (
    nom VARCHAR(255),
    CONSTRAINT PK_GENRE PRIMARY KEY (nom)
);

CREATE TABLE JeuVideo_Genre (
    idJeuVideo SERIAL,
    nomGenre VARCHAR(255),
    CONSTRAINT PK_JeuVideo_Genre PRIMARY KEY (idJeuVideo, nomGenre),
    CONSTRAINT FK_JeuVideo_Genre_idArticle FOREIGN KEY (idJeuVideo)
REFERENCES JeuVideo(idArticle)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_JeuVideo_Genre_nomGenre FOREIGN KEY (nomGenre)
REFERENCES GENRE(nom)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

# Partie 4

- Le client 'YoLo666' veut changer son pseudo en 'BitCoinLover2024'

```sql
UPDATE Client
SET pseudo = 'BitCoinLover2024'
WHERE pseudo = 'YoLo666';
```

La requête est exécutée sans lever une exception

- Ajouter un Client avec 'K3V1N' comme pseudo, le 12 août 2009 comme date de naissance, 'Rue du haut 12, 1004 Lausanne' comme adresse et 'HeadShot2012@gmail.com' comme email.

```
INSERT INTO Client (pseudo, dateNaissance, adresseFacturation, email)
VALUES ('K3V1N', '2009-08-12', 'Rue du haut 12, 1004 Lausanne',
'HeadShot2012@gmail.com');
```

Une exception est levé indiquant qu'il y a un doublon dans le champ « email » car celui-ci contient une contrainte d'unicité

- Supprimer le jeu vidéo 'Demon's Souls' (id 5)

```
DELETE FROM Article
WHERE id = 5;
```

La requête est exécutée sans lever une exception

- Ajouter les mots-clés 'loot' et 'Horadrims' à l'article 'Diablo IV' (id 3)

```
INSERT INTO Article_MotsCles (articleId, motCle)
VALUES
    (3, 'loot'),
    (3, 'Horadrims');
```

La requête est exécutée sans lever une exception

# Partie 5

- L'attribut Extention.necessiteJeuDeBase ait TRUE comme valeur par défaut.

```
ALTER TABLE DLC
ALTER COLUMN necessiteJeuDeBase SET DEFAULT TRUE;
```

La modification peut être implémentée sans problème

- L'âge minimum pour un JeuVideo ne puisse ni être supérieur à 18 ans ni inférieur à 12 ans.

```
ALTER TABLE JeuVideo
ADD CONSTRAINT CK_JeuVideo_ageMinimum CHECK (ageMinimum BETWEEN 12 AND 18);
```

Non, la modification ne peut pas être implémentée directement si des tuples sont hors des plages

- Le nombre de copies restantes de chaque article par console soit stocké.

```
ALTER TABLE Article_Console
ADD copiesRestantes INT NOT NULL DEFAULT 0;
```

La modification peut être implémentée sans problème

- Chaque client ait la possibilité d'acheter plusieurs fois le même article (à des dates différentes).

```
ALTER TABLE Client_Article
DROP CONSTRAINT PK_Client_Article;

ALTER TABLE Client_Article
ADD CONSTRAINT PK_Client_Article PRIMARY KEY (pseudoClient, idArticle,
dateAchat);
```

Non, la modification ne peut pas être implémentée directement si des tuples existants ne respectent pas la nouvelle contrainte

# Partie 6

## Identifier pour quelle console chaque article a été acheté

Pour ce faire on a besoin de créer une table Achat qui contenir les attributs de la table Client_Article + les référence des tables Client, Article et Console. Les tables Client_Article et Article_Console qui ne font plus de sens.

```
CREATE TABLE Achat (
    idAchat SERIAL,
    pseudoClient VARCHAR(80) NOT NULL,
    idArticle SERIAL NOT NULL,
    nomConsole VARCHAR(255),
    dateAchat DATE NOT NULL,
    copiesRestantes INTEGER NOT NULL,
    CONSTRAINT PK_Achat PRIMARY KEY (idAchat),
    CONSTRAINT FK_Achat_Client FOREIGN KEY (pseudoClient) REFERENCES Client(pseudo)
    ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_Achat_Article FOREIGN KEY (idArticle) REFERENCES Article(id)
    ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_Achat_Console FOREIGN KEY (nomConsole) REFERENCES Console(nom)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

## Empêcher l'achat d'un article pour une console sur laquelle il n'est pas disponible

On peut regarder si un article est épuisée en faisant cette requête à l'aide de l'attribut copiesRestantes

```
SELECT copiesRestantes
FROM Achat
WHERE idArticle = 1 AND nomConsole = 'PS5' AND historique = TRUE AND
copiesRestantes > 0;
```

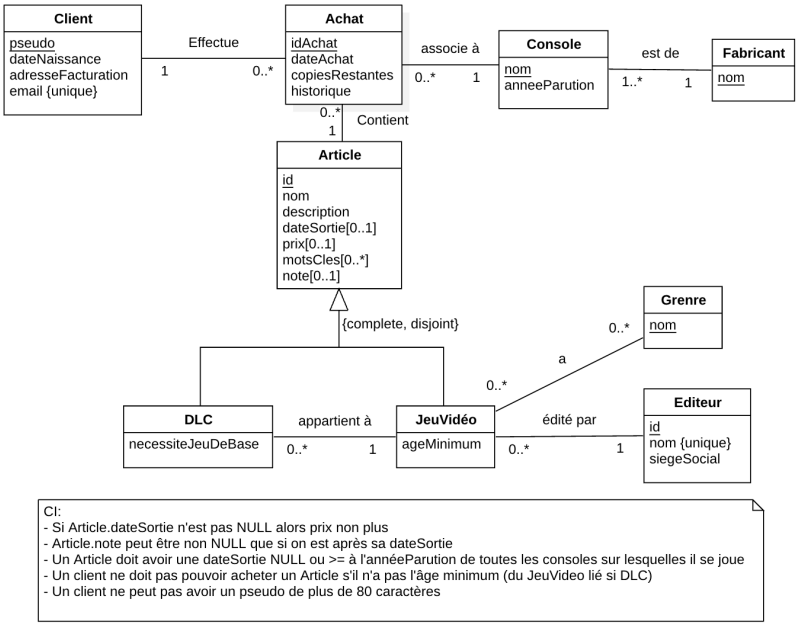
## Conserver les anciennes données d'achat

Nous avons décidé d'ajouter une colonne dans la table Achat de type boolean qui est par défaut FALSE pour permettre de différencier les tuples historiques aux tuples récents.

```
ALTER TABLE Achat
ADD COLUMN historique BOOLEAN DEFAULT FALSE;
```

## Schéma EA modifié

Voici le schéma EA modifié en fonction des besoins demandés. On remarque que les tables d'association Client_Article et Article_Console ont été remplacées par la table Achat où l'on a rajouté les attributs de Client_Article en plus d'un attribut pour gérer les données historiques.




# Conclusion

Ce laboratoire nous a permis d'appliquer les notions apprises en cours, notamment la transformation d'un modèle conceptuel en modèle relationnel, l'écriture de scripts SQL pour créer et manipuler des bases de données, et la gestion des contraintes d'intégrité pour répondre à des besoins concrets.