Rapport

Auteurs: Nathan Füllemann, Mathéo Lopez

Partie 1

Client(pseudo, dateNaissance, adresseFacturation, email)

Client.email UNIQUE

Article(id, nom, description, dateSortie, prix, motsCles, note)

Achat(id, pseudo, dateAchat)

Achat.id reference Article.id

Achat.pseudo reference Client.pseudo

Console(nom, annéeParution, nomFabricant)

Article_Console(id, nom)

Article_Console.nom reference Console.nom

Console.nom NOT NULL

Article_Console.id reference Article.id

Fabricant(nom)

DLC(id, necessiteJeuDeBase, idJeuVideo)

DLC.id reference Article.id

DLC.idJeuVideo reference JeuVideo.id

JeuVideo(id, ageMinimum, idEditeur)

JeuVideo.id reference Article.id

JeuVideo.idEditeur reference Editeur.id

Genre(nom)

JeuVideo_Genre(nom, id)

JeuVideo_Genre.nom reference Genre.nom

JeuVideo_Genre.id reference JeuVideo.id

Editeur(id, nom, SiegeSocial)

Partie 2

```sql
-- Table Client
CREATE TABLE Client (
    client_id INT PRIMARY KEY,
    pseudo VARCHAR(80) NOT NULL, -- Limite de 80 caractères pour le pseudo
    dateNaissance DATE,
    adresseFacturation VARCHAR(255),
    email VARCHAR(255) UNIQUE -- Contrainte d'unicité sur l'email
);

-- Table Fabricant
CREATE TABLE Fabricant (
    fabricant_id INT PRIMARY KEY,
    nom VARCHAR(255) NOT NULL
);

-- Table Console
CREATE TABLE Console (
    console_id INT PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    anneeParution YEAR NOT NULL,
    fabricant_id INT,
    FOREIGN KEY (fabricant_id) REFERENCES Fabricant(fabricant_id) ON DELETE SET
NULL
);

-- Table Editeur
CREATE TABLE Editeur (
    editeur_id INT PRIMARY KEY,
    nom VARCHAR(255) UNIQUE NOT NULL, -- Contrainte d'unicité sur le nom
    siegeSocial VARCHAR(255)
);

-- Table Genre
CREATE TABLE Genre (
    genre_id INT PRIMARY KEY,
    nom VARCHAR(255) NOT NULL
);

-- Table Article
CREATE TABLE Article (
    article_id INT PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    description TEXT,
    dateSortie DATE,
    prix DECIMAL(10, 2) CHECK (prix IS NOT NULL OR dateSortie IS NULL), -- CI:
prix non NULL si dateSortie non NULL
    motsCles TEXT,
    note DECIMAL(3, 1) CHECK (note IS NULL OR dateSortie <= CURRENT_DATE) -- CI:
note non NULL si on est après dateSortie
);

-- Table JeuVideo
```

```sql
CREATE TABLE JeuVideo (
    jeu_id INT PRIMARY KEY,
    article_id INT UNIQUE, -- Association 1:1 avec Article
    ageMinimum INT CHECK (ageMinimum BETWEEN 12 AND 18), -- CI: âge minimum entre
12 et 18 ans
    editeur_id INT,
    FOREIGN KEY (article_id) REFERENCES Article(article_id) ON DELETE CASCADE,
    FOREIGN KEY (editeur_id) REFERENCES Editeur(editeur_id) ON DELETE SET NULL
);

-- Table DLC
CREATE TABLE DLC (
    dlc_id INT PRIMARY KEY,
    article_id INT UNIQUE, -- Association 1:1 avec Article
    necessiteJeuDeBase BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (article_id) REFERENCES Article(article_id) ON DELETE CASCADE
);

-- Table de relation JeuVideo_Genre (association n:m entre JeuVideo et Genre)
CREATE TABLE JeuVideo_Genre (
    jeu_id INT,
    genre_id INT,
    PRIMARY KEY (jeu_id, genre_id),
    FOREIGN KEY (jeu_id) REFERENCES JeuVideo(jeu_id) ON DELETE CASCADE,
    FOREIGN KEY (genre_id) REFERENCES Genre(genre_id) ON DELETE CASCADE
);

-- Table de relation Article_Console (association n:m entre Article et Console)
CREATE TABLE Article_Console (
    article_id INT,
    console_id INT,
    PRIMARY KEY (article_id, console_id),
    FOREIGN KEY (article_id) REFERENCES Article(article_id) ON DELETE CASCADE,
    FOREIGN KEY (console_id) REFERENCES Console(console_id) ON DELETE CASCADE
);

-- Table de relation Client_Article (association n:m entre Client et Article)
CREATE TABLE Client_Article (
    client_id INT,
    article_id INT,
    dateAchat DATE,
    PRIMARY KEY (client_id, article_id, dateAchat),
    FOREIGN KEY (client_id) REFERENCES Client(client_id) ON DELETE CASCADE,
    FOREIGN KEY (article_id) REFERENCES Article(article_id) ON DELETE CASCADE,
    CHECK ((SELECT ageMinimum FROM JeuVideo WHERE JeuVideo.article_id =
article_id) <=
           (SELECT YEAR(CURRENT_DATE) - YEAR(dateNaissance) FROM Client WHERE
Client.client_id = client_id))
    -- CI: âge minimum pour acheter un jeu vidéo respecté
);
```

Partie 3

Le script s'execute correctement et sans erreur.

Partie 4

1) Changer le pseudo du client 'YoLo666' en 'BitCoinLover2024'

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

Explication :

Cette instruction met à jour le pseudo du client dans la table Client. Elle cherche le client ayant le pseudo 'YoLo666' et le remplace par 'BitCoinLover2024'. Si l'opération ne fonctionne pas, cela pourrait être dû à une contrainte d'unicité sur la colonne pseudo (si cette contrainte existe), ce qui empêcherait deux clients d'avoir des pseudos identiques.

2) 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');
```

Explication :

Cette instruction insère un nouveau client avec les informations fournies dans la table Client. Si l'opération échoue, cela peut être dû à une violation de contrainte, comme une duplication de l'email (si une contrainte d'unicité est définie sur la colonne email) ou un format de date incorrect.

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

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

Explication :

Cette instruction supprime l'article correspondant à 'Demon's Souls' dans la table Article à l'aide de son idArticle (5). Cependant, si des contraintes de clé étrangère existent, l'opération peut échouer. Par exemple, si un jeu vidéo est référencé dans la table JeuVideo (table liée à l'article via idArticle), la suppression sera

bloquée. Dans ce cas, il faudrait d'abord supprimer les références dans les autres tables ou désactiver les contraintes de clé étrangère temporairement.

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

```
INSERT INTO MotsCles (idArticle, motCle)
VALUES (3, 'loot'), (3, 'Horadrims');
```

Explication :

Cette instruction ajoute les mots-clés 'loot' et 'Horadrims' à l'article ayant l'id 3 (correspondant à 'Diablo IV') dans la table MotsCles. Si l'opération échoue, cela peut être dû à une violation de contrainte, par exemple, si les mots-clés doivent être uniques dans la base de données ou s'il y a des règles empêchant l'ajout de mots-clés pour cet article.

Partie 5

1) L'attribut `Extention.necessiteJeuDeBase` doit avoir `TRUE` comme valeur par défaut

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

Explication :

Cette instruction modifie la colonne `necessiteJeuDeBase` de la table `Extention` pour que sa valeur par défaut soit `TRUE`. Si l'opération échoue, cela peut être dû à une restriction d'intégrité ou à des dépendances existantes dans la table. Les données déjà présentes dans la base ne seront pas affectées par cette modification de valeur par défaut.

2) L'âge minimum pour un `JeuVideo` ne peut ni être supérieur à 18 ans ni inférieur à 12 ans

```
ALTER TABLE JeuVideo
ADD CONSTRAINT check_age CHECK (ageMinimum >= 12 AND ageMinimum <= 18);
```

Explication :

Cette instruction ajoute une contrainte `CHECK` à la colonne `ageMinimum` de la table `JeuVideo` afin que l'âge minimum soit compris entre 12 et 18 ans. Si des tuples existent déjà dans la table `JeuVideo` avec un âge minimum en dehors de cette plage, l'opération échouera. Il sera donc nécessaire de vérifier et corriger les valeurs de la colonne avant d'appliquer cette contrainte.

3) Le nombre de copies restantes de chaque article par console doit être stocké

```
ALTER TABLE Article_Console
ADD COLUMN copiesRestantes INT DEFAULT 0;
```

Explication :

Cette instruction ajoute une colonne `copiesRestantes` à la table `Article_Console` pour stocker le nombre de copies restantes d'un article pour chaque console. Les valeurs existantes dans cette colonne seront par défaut définies à 0. Cependant, il est important de vérifier que des données valides sont insérées dans cette nouvelle colonne pour chaque paire d'article et de console.

4) Chaque client doit avoir la possibilité d'acheter plusieurs fois le même article (à des dates différentes)

```
ALTER TABLE Client_Article
DROP CONSTRAINT client_article_unique;

ALTER TABLE Client_Article
ADD CONSTRAINT client_article_unique UNIQUE (pseudoClient, idArticle, dateAchat);
```

Explication :

Cette instruction supprime la contrainte d'unicité sur les colonnes `pseudoClient` et `idArticle` dans la table `Client_Article` pour permettre à un client d'acheter plusieurs fois le même article à des dates différentes. Les tuples existants doivent être vérifiés pour s'assurer qu'il n'existe pas de doublons sur ces colonnes avant d'appliquer cette modification.

Partie 6

1) Gestion de l'achat d'articles par console

Pour garantir qu'un article ne puisse être acheté que pour une console sur laquelle il est disponible, il faut modifier la table de jointure `Article_Console`.

Modification proposée :

- **Table `Article_Console`** : Cette table doit être utilisée pour déterminer sur quelles consoles un article est disponible. Chaque `idArticle` devrait être associé uniquement aux consoles sur lesquelles il est disponible.
- **Ajout d'une contrainte d'intégrité référentielle** : La table `Client_Article` devrait être liée à `Article_Console` pour s'assurer qu'un article ne soit acheté que sur une console où il est disponible.

Exemple de modification :

```

```

```
ALTER TABLE Client_Article
ADD CONSTRAINT fk_article_console
FOREIGN KEY (idArticle, nomConsole) REFERENCES Article_Console(idArticle,
nomConsole);
```

2) Stockage des anciens achats de clients sans information sur la console

Le magasin souhaite conserver l'historique des achats sans connaître la console concernée pour les anciens achats. Pour cela, nous allons créer une table distincte pour enregistrer les achats passés.

Solution proposée :

- **Table** `Ancien_Achat` : Cette table contiendra les informations des achats passés, sans référence à la console.

Exemple de création de table :

```
CREATE TABLE Ancien_Achat (
  idAncienAchat INT PRIMARY KEY AUTO_INCREMENT,
  pseudoClient VARCHAR(80),
  idArticle INT,
  dateAchat DATE,
  FOREIGN KEY (pseudoClient) REFERENCES Client(pseudo),
  FOREIGN KEY (idArticle) REFERENCES Article(idArticle)
);
```

Cela permettra de conserver l'historique des achats sans l'information de la console.

3) Réflexion sur d'autres attributs ou associations à déplacer

Il peut être nécessaire de déplacer certains attributs ou associations dans la modélisation pour garantir la cohérence des données et éviter des incohérences dues à ce refactoring.

Modifications proposées :

- **Déplacement de l'attribut** `copiesRestantes` : Cet attribut devrait être déplacé dans une nouvelle table de gestion des stocks, par exemple `Stock_Article_Console`, afin de mieux gérer les stocks pour chaque article par console.

Exemple de nouvelle table pour les stocks :

```
CREATE TABLE Stock_Article_Console (
  idArticle INT,
  nomConsole VARCHAR(255),
  copiesRestantes INT DEFAULT 0,
  PRIMARY KEY (idArticle, nomConsole),
```

```
    FOREIGN KEY (idArticle) REFERENCES Article(idArticle),
    FOREIGN KEY (nomConsole) REFERENCES Console(nom)
);
```

- **Ajout de `dateSortie` dans `Ancien_Achat`** : Pour garder une trace de la date de sortie de l'article au moment de l'achat, nous pouvons ajouter une colonne `dateSortie` dans la table `Ancien_Achat`, bien que ce ne soit pas obligatoire.

4) Adaptation du schéma EA dans StarUML

Les modifications suivantes doivent être appliquées au diagramme EA dans StarUML :

1. **Ajout de la table `Ancien_Achat`** pour les anciens achats.
2. **Modification de `Article_Console`** pour garantir l'intégrité des achats par console.
3. **Ajout de la table `Stock_Article_Console`** pour gérer les copies restantes d'articles par console.

Relations à modifier :

- **Client** - [1, N] -> **Client_Article** [1, N] -> **Article** [1, N] -> **Console**
- **Ancien_Achat** [1, N] -> **Article**
- **Article_Console** [1, N] -> **Console**
- **Stock_Article_Console** [1, N] -> **Console**