

Compte rendu TP 2 bases de données

Par : ayoub zarouati - Master II-BDCC

Dans le cadre de ce compte rendu, nous avons été amenés à concevoir et mettre en œuvre un Modèle Logique des Données (MLD) pour une plate-forme de streaming vidéo. Ce modèle vise à structurer les informations liées aux utilisateurs, aux films, aux abonnements, ainsi qu'à l'historique de visionnage.

Le travail réalisé inclut la modélisation des tables suivantes :

1. **Utilisateur** : pour enregistrer les informations de base des utilisateurs inscrits sur la plateforme.
2. **Film** : pour stocker les informations relatives aux films disponibles.
3. **Abonnement** : pour gérer les différents types d'abonnements souscrits par les utilisateurs.
4. **Historique de Visionnage** : pour suivre les films visionnés par les utilisateurs.

La structure de chaque table est décrite en termes de colonnes et de types de données, avec l'établissement des clés primaires et des relations entre les tables. Ensuite, des exemples d'insertion de données ont été fournis pour chacune des tables, permettant ainsi de simuler des scénarios d'utilisation concrets.

Enfin, plusieurs requêtes SQL ont été proposées pour illustrer la manière dont les informations peuvent être extraites de la base de données afin de répondre à des besoins spécifiques, comme la liste des films visionnés par un utilisateur ou la gestion des abonnements.

Le contenu détaillé de la modélisation, ainsi que les requêtes SQL, est présenté ci-après.

Modèle Logique des Données (MLD):

· Utilisateur

Utilisateur(id_utilisateur, nom, email, date_inscription)

· Film

Film(id_film, titre, genre, annee_sortie, duree)

- **Abonnement**

```
Abonnement(id_abonnement, type_abonnement, date_debut, date_fin,  
#id_utilisateur)
```

- **Historique_Visionnage**

```
Historique_Visionnage(id_historique, date_visionnage, #id_utilisateur, #id_film)
```

La création de la base de données / tables:

Create database db_streaming

```
-- Table Utilisateur
```

```
CREATE TABLE Utilisateur ( id_utilisateur INT AUTO_INCREMENT PRIMARY KEY, nom VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL UNIQUE, date_inscription DATE NOT NULL);
```

```
-- Table Film
```

```
CREATE TABLE Film ( id_film INT AUTO_INCREMENT PRIMARY KEY, titre VARCHAR(255) NOT NULL, genre VARCHAR(100) NOT NULL, annee_sortie YEAR NOT NULL, duree INT NOT NULL);
```

```
-- Table Abonnement
```

```
CREATE TABLE Abonnement ( id_abonnement INT AUTO_INCREMENT PRIMARY KEY, id_utilisateur INT NOT NULL, type_abonnement ENUM('Basic', 'Standard', 'Premium') NOT NULL, date_debut DATE NOT NULL, date_fin DATE NOT NULL, FOREIGN KEY (id_utilisateur) REFERENCES Utilisateur(id_utilisateur));
```

```
-- Table Historique_Visionnage
```

```
CREATE TABLE Historique_Visionnage ( id_historique INT AUTO_INCREMENT PRIMARY KEY, id_utilisateur INT NOT NULL, id_film INT NOT NULL, date_visionnage DATE NOT NULL, FOREIGN KEY (id_utilisateur) REFERENCES Utilisateur(id_utilisateur), FOREIGN KEY (id_film) REFERENCES Film(id_film));
```

Insertion de données:

Exemples d'insertion de données dans les tables:

```
-- Insertion de données dans la table Utilisateur
```

```
INSERT INTO Utilisateur (nom, email, date_inscription) VALUES ('Alice Dupont', 'alice@example.com', '2023-01-15'), ('Bob Martin', 'bob@example.com', '2023-02-20'), ('Clara Lopez', 'clara@example.com', '2023-03-10');
```

-- Insertion de données dans la table Film

```
INSERT INTO Film (titre, genre, annee_sortie, duree) VALUES ('Inception', 'Science Fiction', 2010, 148), ('The Matrix', 'Action', 1999, 136), ('Interstellar', 'Science Fiction', 2014, 169);
```

-- Insertion de données dans la table Abonnement

```
INSERT INTO Abonnement (id_abonnement, type_abonnement, date_debut, date_fin, id_utilisateur) VALUES (1, 'Premium', '2023-01-15', '2024-01-14', 7), (2, 'Standard', '2023-02-20', '2024-02-19', 8), (3, 'Basic', '2023-03-10', '2024-03-09', 9);
```

-- Insertion de données dans la table Historique_Visionnage

```
INSERT INTO Historique_Visionnage (id_utilisateur, id_film, date_visionnage) VALUES (7, 1, '2023-12-01'), (8, 2, '2023-12-05'), (9, 3, '2023-12-10');
```

Partie 2:

1. Listez tous les utilisateurs inscrits depuis plus d'un an.

```
SELECT nom, email, date_inscription FROM Utilisateur WHERE DATEDIFF(CURRENT_DATE, date_inscription) > 365;
```

2. Affichez les films d'un genre spécifique, triés par année de sortie.

```
SELECT titre, genre, annee_sortie FROM Film WHERE genre = 'Science Fiction' ORDER BY annee_sortie DESC;
```

3. Trouvez les utilisateurs ayant un abonnement 'Premium' dont la date de fin est dans

moins d'un mois.

```
SELECT u.nom, u.email, a.date_fin FROM Utilisateur u JOIN Abonnement a ON u.id_utilisateur = a.id_utilisateur WHERE a.type_abonnement = 'Premium' AND DATEDIFF(a.date_fin, CURRENT_DATE) <= 30;
```

nom	email	date_fin
Alice Dupont	alice@example.com	2024-01-14

4. Affichez les utilisateurs et les films qu'ils ont visionnés au cours de la dernière semaine.

```
SELECT u.nom, f.titre, h.date_visionnage FROM Utilisateur u JOIN Historique_Visionnage h ON u.id_utilisateur = h.id_utilisateur JOIN Film f ON h.id_film = f.id_film WHERE h.date_visionnage >= DATE_SUB(CURRENT_DATE, INTERVAL 7 DAY);
```

5. Listez les films qui n'ont pas été visionnés par aucun utilisateur.

```
SELECT titre FROM Film f LEFT JOIN Historique_Visionnage h ON f.id_film = h.id_film WHERE h.id_historique IS NULL;
```

6. Trouvez les utilisateurs qui ont regardé plus de 10 films depuis leur inscription.

```
SELECT u.nom, COUNT(h.id_historique) AS nb_films FROM Utilisateur u JOIN Historique_Visionnage h ON u.id_utilisateur = h.id_utilisateur GROUP BY u.nom HAVING COUNT(h.id_historique) > 10;
```

7. Affichez tous les abonnements, y compris ceux des utilisateurs non enregistrés dans la table des visionnages.

```
SELECT u.nom, a.type_abonnement, a.date_debut, a.date_fin FROM Abonnement a LEFT JOIN Utilisateur u ON a.id_utilisateur = u.id_utilisateur;
```

8. Affichez tous les utilisateurs, même ceux qui n'ont pas d'abonnement.

```
SELECT u.nom, a.type_abonnement FROM Utilisateur u LEFT JOIN Abonnement a ON u.id_utilisateur = a.id_utilisateur;
```

9. Listez les utilisateurs qui ont regardé le même film plusieurs fois.

```
SELECT u.nom, f.titre, COUNT(h.id_historique) AS nb_fois FROM Utilisateur u JOIN Historique_Visionnage h ON u.id_utilisateur = h.id_utilisateur JOIN Film f ON h.id_film = f.id_film GROUP BY u.nom, f.titre HAVING COUNT(h.id_historique) > 1;
```

10. Trouvez les 3 films les plus regardés au cours des 6 derniers mois.

```
SELECT f.titre, COUNT(h.id_historique) AS nb_visionnages FROM Film f JOIN Historique_Visionnage h ON f.id_film = h.id_film WHERE h.date_visionnage >= DATE_SUB(CURRENT_DATE, INTERVAL 6 MONTH) GROUP BY f.titre ORDER BY nb_visionnages DESC LIMIT 3;
```

11. Identifiez les utilisateurs ayant un abonnement actif, mais n'ayant visionné aucun film.

```
SELECT u.nom, a.type_abonnement FROM Utilisateur u JOIN Abonnement a ON u.id_utilisateur = a.id_utilisateur LEFT JOIN Historique_Visionnage h ON u.id_utilisateur = h.id_utilisateur WHERE h.id_historique IS NULL AND a.date_fin >= CURRENT_DATE;
```

12. Créez un utilisateur SQL avec un accès en lecture seule sur toutes les tables.

```
CREATE USER 'read_only_user'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT SELECT ON *.* TO 'read_only_user'@'localhost';
```

13. Accordez à un administrateur SQL le privilège de modifier les informations des films et des abonnements.

```
GRANT UPDATE, DELETE, INSERT ON Film TO 'admin_user'@'localhost';
```

```
GRANT UPDATE, DELETE, INSERT ON Abonnement TO 'admin_user'@'localhost';
```

14. Affichez les films qui durent plus de 2 heures et ont été visionnés par des utilisateurs ayant un abonnement 'Standard'.

```
SELECT f.titre, f.duree, u.nom FROM Film f JOIN Historique_Visionnage h ON f.id_film = h.id_film JOIN Utilisateur u ON h.id_utilisateur = u.id_utilisateur JOIN Abonnement a ON u.id_utilisateur = a.id_utilisateur WHERE f.duree > 120 AND a.type_abonnement = 'Standard';
```

titre	duree	nom
The Matrix	136	Bob Martin

15. Trouvez les utilisateurs ayant regardé un film d'un genre spécifique, mais n'ayant pas d'abonnement 'Premium'.

```
SELECT u.nom, f.titre, f.genre FROM Utilisateur u JOIN Historique_Visionnage h ON u.id_utilisateur = h.id_utilisateur JOIN Film f ON h.id_film = f.id_film JOIN Abonnement a ON u.id_utilisateur = a.id_utilisateur WHERE f.genre = 'Action' -- ou un autre genre spécifique AND a.type_abonnement != 'Premium';
```

nom	titre	genre
Bob Martin	The Matrix	Action

Fin de document