École nationale supérieure d'électronique, informatique,
télécommunications, mathématique et mécanique de Bordeaux

Rapport de projet Base de Données Jeux-Vidéos

Trinôme : mchatt, qliang et mremmach

Table des matières

Introduction

L'information est aujourd'hui le principal acteur dans tous les domaines. L'informatique, plus que tous autre discipline repose et utilise massivement des données. Par soucis d'organisation et d'efficacité il est recommandé d'utiliser un système de gestion de base de données. Ce système permet d'organiser les données d'une manière facilitant leur exploitation.
Les systèmes de gestion de base données permettent de stocker les informations sous forme de table et de relier ces tables entre-elles. Cette propriété est très pratique pour mieux schématiser et modéliser les données utilisées et une bonne implémentation de la base permet d'éviter toute redondance de données ou duplication d'informations.
Ce projet se propose de réaliser une base de données représentant la gestion d'une communauté de joueurs de jeux-vidéos. La conception de cette base passe d'abord par une modélisation conceptuelle et relationnelle, ensuite, par l'écriture de requêtes pour pouvoir exploiter les données et enfin, par la mise en place d'une interface pour faciliter l'interaction avec l'utilisateur.

1 Modélisation

Dans cette partie nous présenterons les différentes refléxions faites sur le modèle de la base ainsi que le shéma retenu.

1.1 Hypothèses

1.1.1 Table de données

Pour définir l'ensemble des tables qui vont composer la base, il est nécessaire de définir avec précision les différentes entités qui interviennent ainsi que les différents attributs qui les caractérisent. Les attributs pouvant être déduits des autres n'apparaissent pas. Cette analyse permet d'avoir une vision globale des données stockées.

Joueur
Id_Joueur
Pseudo_Joueur
Nom_Joueur
Prenom_Joueur
Email_Joueur
Password

Le site communautaire permet à ses membres de se regrouper autour d'une thématique qui leur est commune. Chaque joueur possède un `Id_Joueur`, un `Pseudo` et un `Password`. Il indique également son nom, prénom et email.

Le site permet à des joueurs de noter et commenter des jeux. Pour leur permettre de se retrouver, la table jeu contient l'`Id_Jeu`, le `Nom_Jeu`, la `Date_Parution` ainsi qu'une `image` (jaquette) du jeu.

Jeu
Id_Jeu
Nom_Jeu
Date_Parution
Image

Commentaire
Id_Commentaire
Contenu

Chaque Joueur peut peut noter/commenter un jeu ainsi que juger de la pertinence des commentaires d'autres joueurs. Chaque commentaire possède un `Id_Commentaire` et un `Contenu` texte.

Les jeux sont édités par des éditeurs se caractérisant uniquement par leur nom. La table editeur contient donc l'id et le nom de chaque éditeur.

Editeur
Id_Editeur
Nom_Editeur

Categorie
Id_Categorie
Nom_Categorie

" Une catégorie de jeu vidéo désigne un ensemble de jeux vidéo caractérisé par un gameplay similaire. " La table Catégorie contient alors l'id et le nom de chaque catégorie.

Plateforme
Id_Plateforme
Nom_Plateforme

Depuis l'essor du jeu vidéo, différentes consoles concurrentes ont été mise sur le marché et même les ordinateurs peuvent être utilisés comme tel (ils peuvent même être plus puissant). Un même jeu est souvent disponible sur plusieurs plates-formes à la fois. La table Plateforme contient donc l'id et le nom de chaque plates-formes.

1.1.2 Table des relations

Note
Id_Note
Note
Date_Note

Chaque joueur a la possibilité de noter un jeu. Cette note sera comprise entre 0 et 20. Pour avoir la possibilité d'effectuer des statistiques, la date de la note sera enregistrée.

Pertinence
Valeur

Chaque joueur peut indiquer s'il juge pertinent ou non un commentaire qui n'est pas le sien, la table Pertinence contient un champs `Valeur` (1 ou -1) pour indiquer cela.

Categories_Jeu
Id_Jeu
Id_Catégorie

L'association Categories_Jeu associe un jeu à une ou plusieurs catégories. La table contient donc `Id_Jeu` et `Id_Categorie`.

Cette association permet de relier jeu et éditeur.

Edité par

Disponible

La table Disponible est une association entre les deux entités Jeu et Plateforme. Un jeu peut être disponible sur plusieurs plate-formes à la fois, alors qu'une plate-forme peut être liée à des jeux ou pas.

Categorie_Préférée

Chaque joueur indique à son inscription ses préférences, une catégorie et une plate-forme. Cela permet par la suite d'affichier à un joueur les jeux disponibles dans sa catégorie ou sa plate-forme préférée.

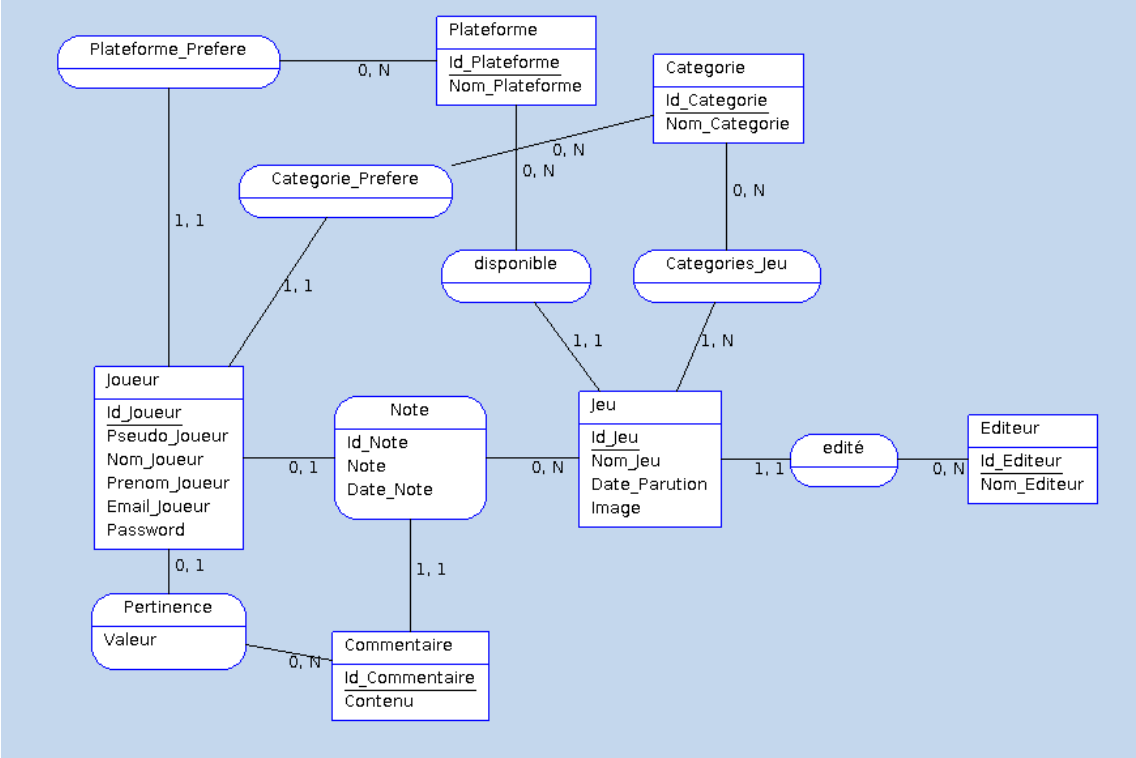
Plateforme_Préférée

1.1.3 Schéma Entité-Association

A partir de la définition des tables, nous avons élaboré le schéma entité-relation suivant. Ce schéma synthétise l'organisation et la manière dont les données sont stockées dans la base. Il permet aussi de

préciser les cardinalités qui donnent des renseignements sur le minimum et le maximum d'occurrences d'une association entre deux entités, par exemple un jeu ne peut être édité que par un et un seul éditeur, alors qu'un éditeur peut éditer un nombre indéfini de 0 à n jeux.

FIGURE 1 – Schéma Entité-Relation



1.2 Modèle Relationnel

1.2.1 Schéma relationnel

Si les données ont été organisé grâce au schéma précédent, elles ne sont pas encore exploitables et ne peuvent répondre aux demandes du projet. On introduit alors la notion de clé primaire et étrangère. Certaines clés primaire sont naturellement associés à des attributs et la seule connaissance de cet attribut permet d'avoir toute les informations sur cette entrée de la table. Toutefois pour faciliter les opérations, nous avons placé dans toutes les entités un champ `ID` qui aura pour seule fonction de servir comme clé primaire.
Dans le cas de la table Pertinence ainsi que de Categories_Jeu, deux attributs permettent de définir une entrée, ces couples sont des clés étrangères et permettent de relier deux entités.
Le schéma conceptuel suivant permet de résumer les relations de clés primaires et clés étrangères. Chaque table est décrite par tous ses attributs. Les clés primaires sont soulignées et les clés étrangères précédées par #.

Joueur	(Id_Joueur, Pseudo_Joueur, Nom_Joueur, Prenom_Joueur, Email_Joueur, #Id_Categorie, #Id_Plateforme, password)
Jeu	(Id_Jeu, Nom_Jeu, Date_Parution, #Id_Editeur, #Id_Plateforme, img)
Editeur	(Id_Editeur, Nom_Editeur)
Plateforme	(Id_Plateforme, Nom_Plateforme)
Categorie	(Id_Categorie,Nom_Categorie)
Categories_Jeu	(#Id_Categorie,#Id_Jeu)
Note	(Id_Note, #Id_Joueur, #Id_Jeu, #Id_Commentaire, Note, Date_Note)
Commentaire	(Id_Commentaire, Contenu)
Pertinence	(#Id_Commentaire, #Id_Joueur, Valeur)

TABLE 1 – Schéma Conceptuel

1.2.2 Description et Hypothèses

Lors de la conception de notre base de données, nous avons tâché de respecter les 3 premières formes normales. Les 2 premières formes normales sont respectés.
- Les attributs de chaque table sont atomiques.
- Un attribut non clé primaire ne doit pas dépendre d'une partie de la clé primaire. Il doit en dépendre entièrement.

La 3ème forme normale est quant à elle respéctée que partiellement. Cette forme stipule que tout attribut non clé ne dépend pas d'un ou plusieurs attributs ne participant pas à la clé. Dans la table Joueur, l'Id_Joueur est la seule clé primaire, mais nous avons par la suite déclaré les attributs Pseudo et Email comme unique, ils peuvent ainsi avoir le rôle de clé primaire et donc le Nom ou le Prenom peuvent directement dépendre du Pseudo. Toutefois elle reste respecté sur toutes les autres tables autres que Joueur.

2 Implémentation

Pour l'implémentation de la base, nous avons choisi d'utiliser le duo PHP/Mysql.
Le choix de Mysql en tant que SGBD pour le projet est justifié par plusieurs arguments :
- Il est gratuit et donc accessible.
- Utilisation facile et syntaxe claire.
- Assez puissant pour les besoins du projet.

Les raisons justifiant le choix de PHP comme langage pour l'interface sont assez similaires à celles évoquées pour Mysql. Nous pouvons y ajouter le fait que PHP offre une interface simple et puissante dénommée PDO qui permet de se connecter à la plupart des SGBD connus de la même manière. Par contre il faudra bien faire attention à respecter la syntaxe de chaque SGBD. De plus le couple PHP/Mysql est très répandu, ce qui atteste de sa fiabilité.

2.1 Implémentation de la base

La base est implémenté à travers des requêtes répartis sur plusieurs fichiers.

2.1.1 Création

La base est crée à l'aide du fichier *base.sql*. On commence par supprimer les tables qui portent le même nom que celles qu'on veut créer (si elles existent). Ceci permet d'éviter tout conflit potentiel avec des données existantes dans la base. On crée ensuite les tables qui constituent la base en précisant les propriétés de chacune d'entre elles. Par exemple : le type de chaque attribut, la clé primaire et l'auto

incrémentation pour la clé primaire ...
Après avoir crée les tables, on ajoute les contraintes sur les attributs afin de garantir la cohérence de la base et de respecter la modélisation que nous avons faite. Nous avons, par ailleurs, choisi de mettre des déclencheurs (présentés plus bas) dans ce même fichier car ils permettent d'assurer la cohérence de certains éléments de la base et font donc partie du script de création.

2.1.2 Suppression

Les requêtes permettant de supprimer la base sont présentées dans le fichier *suppr.sql*, Elles sont les mêmes qu'au début du fichier *base.sql*. Toutefois il faut prendre garde à ne pas modifier l'ordre des requêtes , sinon certaines ne s' exécuteront pas en renvoyant une erreur de violation des contraintes d'intégrité.

2.1.3 Jeu de données

Pou tester la base et y exécuter les différentes requêtes demandées il a fallu créer un jeu de données pertinent et varié. Pour cela nous avons créé un fichier *donnees.sql* regroupant plusieurs requêtes pour remplir les différentes tables de la base.
Les données ont été mise dans un fichier à part car il est volumineux. De plus cette manière de procéder permet d'obtenir plus de lisibilité et de séparer les tâches entres les fichiers.

2.2 Implémentation des requêtes

Le sujet demande l'exécution d'un certain nombre d'opérations de différent type sur la base. Les requêtes ont été rendu paramétrable pour garantir un maximum de réétulisabilité, ainsi il faudra remplacer les paramètres par des valeurs adéquates (généralement représenté par des « ? »).
Nous allons ici présenté l'ensemble des requêtes demandées.

2.2.1 Requêtes de consultation

Nous présentons ici les requêtes relatives à la consultation des éléments de la base données.

Liste des jeux : Par défaut sur la page des jeux on peut accéder à la liste des jeux classé par moyenne pondéré des notes des joueurs. La pondération des notes se fait en multipliant chaque note attribué par un joueur par son indice de confiance . Enfin on fait la somme des notes pondérées puis on divise par la somme des indices de confiance.

```
SELECT *
FROM ((
          (SELECT Jeu.*,
                  Moy_Pond
           FROM Jeu, (
                      (SELECT Id_Jeu,
                              sum(Note*confiance)/sum(confiance) AS
                                  Moy_Pond
                       FROM (
                         (SELECT Id_Jeu, Id_Commentaire, Note,(1+
                             sum(positif))/(1+sum(negatif)) AS
                              confiance
                          FROM ((
                                  (SELECT Note.Id_Jeu,
                                          Note.Id_Commentaire,
                                          Note.Note ,
                                          count(*) AS negatif,
```

```
                                                    0 AS positif
                                                FROM Pertinence,
                                                     Note
                                                WHERE Note.Id_Commentaire =
                                                    Pertinence.Id_Commentaire
                                                  AND Valeur = -1
                                                GROUP BY Note.Id_Commentaire)
                                            UNION
                                               (SELECT Note.Id_Jeu,
                                                       Note.Id_Commentaire,
                                                       Note.Note ,
                                                       0 AS negatif ,
                                                       count(*) AS positif
                                                FROM Pertinence,
                                                     Note
                                                WHERE Note.Id_Commentaire =
                                                    Pertinence.Id_Commentaire
                                                  AND Valeur = 1
                                                GROUP BY Note.Id_Commentaire))
                                                     AS T)
                                    GROUP BY Id_Commentaire) AS P)
                            GROUP BY Id_Jeu
                            ORDER BY Moy_Pond DESC) AS Y)
            WHERE Jeu.Id_Jeu = Y.Id_Jeu)
          UNION
            (SELECT Jeu.*,
                    0 AS Moy_Pond
            FROM Jeu
            WHERE Id_Jeu NOT IN
                 (SELECT Id_Jeu
                  FROM Note)
            ORDER BY Id_Jeu ASC)) AS Z)
ORDER BY Moy_Pond DESC
```

On peux aussi choisir une plate-forme, et dans ce cas on obtient la liste des jeux disponibles sur cette plate-forme classé par catégorie à l'aide de la requête suivante.

```
SELECT DISTINCT T.*
FROM Categories_Jeu, (
                            (SELECT Jeu.*
                             FROM Jeu,
                                  Note
                             WHERE Jeu.Id_Plateforme = 1
                               AND Jeu.Id_Jeu = Note.Id_Jeu) AS T)
WHERE T.Id_Jeu = Categories_Jeu.Id_Jeu
ORDER BY Categories_Jeu.Id_Categorie;
```

Liste des commentaires : Pour la liste des commentaires d'un jeu dans la catégorie préférée d'un joueur disponible sur sa plate-forme préférée, on procède en deux pas car il se peut qu'il y ait plusieurs jeux disponibles dans la catégorie et plate-forme préférée du joueur.
Premièrement on propose au joueur de consulter l'ensemble des jeux critiqués disponible sur sa catégorie préférée et sa plate-forme préférée avec la requête suivante :

```
SELECT distinct J.Id_Jeu, J.Nom_Jeu, J.Id_Plateforme, J.Id_Editeur, J.img
    , J.Date_Parution
FROM Jeu J, Categories_Jeu CJ, Note N
WHERE J.Id_Plateforme = :plateforme AND CJ.Id_Categorie = :categorie AND
    CJ.Id_Jeu = J.Id_Jeu AND N.Id_Jeu = J.Id_Jeu ORDER BY CJ.Id_Categorie;
```

Une fois le jeu choisi on affiche la liste des notes correspondantes classé par indice de confiance avec la requête suivante :

```
  (SELECT Note.*, (1 + pos) / (1 + neg) AS conf
   FROM Note, (
                ( SELECT Id_Commentaire,
                         sum(T.positif) AS pos,
                         sum(T.negatif) AS neg
                  FROM ((
                          ( SELECT Id_Commentaire,
                                   count(*) AS negatif,
                                   0 AS positif
                            FROM Pertinence
                            WHERE Id_Commentaire IN
                                 ( SELECT Id_Commentaire
                                   FROM Note
                                   WHERE Id_Jeu =?)
                              AND Valeur = -1
                            GROUP BY Id_Commentaire)
                        UNION
                          ( SELECT Id_Commentaire,
                                   0 AS negatif,
                                   count(*) AS positif
                            FROM Pertinence
                            WHERE Id_Commentaire IN
                                 ( SELECT Id_Commentaire
                                   FROM Note
                                   WHERE Id_Jeu =?)
                              AND Valeur =1
                            GROUP BY Id_Commentaire)) AS T)
                  GROUP BY Id_Commentaire) AS F)
   WHERE Note.Id_Jeu =?
     AND Note.Id_Commentaire = F.Id_Commentaire
   ORDER BY (1 + pos) / (1 + neg) DESC)
UNION
  ( SELECT Note.*,
           1 AS conf
   FROM Note
   WHERE Note.Id_Jeu =?
     AND Note.Id_Commentaire NOT IN
       ( SELECT Id_Commentaire
         FROM Pertinence))
ORDER BY conf DESC
```

Liste des joueurs qui ont apprécié un commentaire Dans la réalisation de cette requête, nous divisons la liste en deux partie. Une partie concerne ceux qui ont aimé et ceux qui n'ont pas aimé le

commentaire. Cela laisse à l'utilisateur la liberté de choisir l'information qui l'intéresse.

```
select * from Pertinence where Id_Commentaire = ? and Valeur = ? ;
```

2.2.2 Requêtes statistiques

Les joueurs classés par nombre de jeux qu'ils ont commenté : Les joueurs n'ayant noté aucun jeu sont placés en dernier.

```
(SELECT *
  FROM (
          (SELECT Joueur.*,
                  count(*) AS nb
           FROM Joueur,
                Note
           WHERE Joueur.Id_Joueur = Note.Id_Joueur
           GROUP BY Pseudo_Joueur
           ORDER BY nb ASC)
        UNION
          (SELECT Joueur.*,
                  0 AS nb
           FROM Joueur
           WHERE Id_Joueur NOT IN
                (SELECT Id_Joueur
                 FROM Note))) AS T
  ORDER BY nb DESC);
```

N derniers commentaires : Les commentaires sont ici classés selon leur date par ordre décroissants, la requête affiche que N commentaires, le N est passé en paramètre à `LIMIT`.

```
SELECT *
FROM Note
ORDER BY Date_Note DESC
LIMIT ?
```

Le commentaire le plus populaire : Cette requête compte le nombre d'apparitions de chaque commentaire dans la table Pertinence et renvoie celui ayant le maximum d'apparitions.

```
SELECT Id_Commentaire,
       count(Id_Commentaire)
FROM Pertinence
GROUP BY Id_Commentaire HAVING count(Id_Commentaire) =
  (SELECT max(T.nbs)
   FROM (
          (SELECT count(Id_Commentaire) AS nbs
           FROM Pertinence
           GROUP BY Id_Commentaire) AS T)) LIMIT 1;
```

Les commentaires classés par indice de confiance : Les commentaires sont classés selon leur indice de confiance $\frac{1+c}{1+d}$. Les commentaires n'apparaissant pas dans la table Pertinence ont donc un indice égal à 1.

```
(SELECT Note.*,
        N.confiance AS conf
 FROM Note, (
                (SELECT Id_Commentaire, (1+P.pos)/(1+.P.neg) AS confiance
                 FROM (
                          (SELECT Id_Commentaire,
                                  sum(T.positif) AS pos,
                                  sum(T.negatif) AS neg
                           FROM ((
                                    (SELECT Id_Commentaire,
                                            count(Valeur) AS positif,
                                            0 AS negatif
                                     FROM Pertinence
                                     WHERE Valeur = 1
                                     GROUP BY Id_Commentaire)
                                  UNION
                                    (SELECT Id_Commentaire,
                                            0 AS positif,
                                            count(Valeur) AS negatif
                                     FROM Pertinence
                                     WHERE Valeur = -1
                                     GROUP BY Id_Commentaire)) AS T)
                           GROUP BY Id_Commentaire) AS P)) AS N)
 WHERE Note.Id_Commentaire = N.Id_Commentaire)
UNION
 (SELECT Note.* ,
         1 AS conf
 FROM Note
 WHERE Id_Commentaire NOT IN
      (SELECT Id_Commentaire
       FROM Pertinence))
ORDER BY conf DESC;
```

2.2.3 Requêtes de mise à jour

En sois les requêtes de mise à jour ne sont pas difficiles à écrire, cependant elles peuvent entraîner une grande décohérence et altérer l'intégrité de la base. Nous présenterons, ici, les principales requêtes de mise à jour.

Ajout

– d'une note et d'un commentaire :

```
INSERT INTO `Commentaire` (`Id_Commentaire`, `Contenu`) values('','un_
    commentaire');
-- puis avec l'id du commentaire, on cree une note
INSERT INTO `Note` (`Id_Note`, `Id_Joueur`, `Id_Jeu`, `Id_Commentaire`,
     `Note`, `Date_Note`) VALUES
('','1','1','1','20',now());
```

– d'un jeu :

```
INSERT INTO `Jeu` (`Id_Jeu`, `Nom_Jeu`, `Date_Parution`, `Id_Editeur`,
    `Id_Plateforme`, `img`) VALUES ('',?,?,?,?,?);
```

– d'un joeur :

```
INSERT INTO `Joueur` (`Id_Joueur`, `Pseudo_Joueur`, `Nom_Joueur`, `
    Prenom_Joueur`, `Email_Joueur`, `Id_Plateforme`, `Id_Categorie`, `
    password`) VALUES ('',?,?,?,?,?,?,?);
```

Modification

– d'un commentaire (la note n'est pas modifiable) :

```
update Commentaire set Contenu = ? where Id_Commentaire = ?;
```

– d'un jeu :

```
update Jeu set Nom_Jeu = ?, Date_Parution = ?, Id_Editeur = ?,
    Id_Plateforme = ? ,img = ? where Id_Jeu = ?;
```

la catégorie n'étant pas directement mentionné dans la table jeu en supprime les anciennes catégorie du jeu et on en crée des nouvelles lors de la modification dans la table Categories_Jeu.

– d'un joueur :

```
update Joueur set Nom_Joueur = ?, Pseudo_joueur = ?, Prenom_Joueur = ?,
     Email_Joueur = ?, Id_Plateforme = ?, Id_Categorie = ? where
    Id_joueur = ?
```

Suppression

– d'une note (avec son commentaire et ses pertinences) :

```
delete from Note where Id_Note = ?
```

– d'un joeur (supprimera aussi ses notes et ses appréciation des commentaires) :

```
delete from Joueur where Id_Joueur = ?
```

– d'un jeu (supprimera aussi les notes associées) :

```
delete from Jeu where Id_Jeu = ?
```

2.3 Intégrité et cohérence de la base

Pour assurer l'intégrité de la bases après suppression d'une donnée contenant des contraintes de clé étrangères, nous avons utilisé deux outils. Le premier est la possibilité d'ajouter `ON DELETE CASCADE` lors de la définition des contraintes, cette option permet par exemple de supprimer toutes les pertinences d'un joueur lors sa suppression. Nous avons remarqué que même en mettant cette option sur toutes les contraintes, il restait des données dans la base, les suppressions s'arrêtaient au bout de la première contrainte rencontrée.
Le deuxième outil utilisé nous permettant de résoudre ce problème est le déclencheur. Nous en avons utilisé trois au cours de notre projet. L'exemple suivant par exemple montre un déclencheur qui s'active avant la suppression d'une note : il procède à la suppression de toutes les entrées de la table pertinence ayant le même champ Id_Commentaire et ensuite supprime le commentaire en question avant de supprimer la note souhaité.

```
-- trigger supp note => supp pertinence ==> supp commentaire
delimiter //
CREATE TRIGGER supp_Note BEFORE DELETE ON Note
        FOR EACH ROW
        BEGIN
        DELETE from Commentaire WHERE Commentaire.Id_Commentaire = OLD.
            Id_Commentaire;
        END //
delimiter ;
```

2.4 Interface PHP

L'interface PHP permet d'interagir intuitivement avec la base. Elle exécute les requêtes de l'utilisateur et lui renvoie le résultat. Pour bien gérer l'interface les tables de la base ont été modélisé par des classes. Les tables principales de la bases sont représentées par une classe qui porte leur nom et qui contient les informations sur ces dernières. Il y'a aussi une autre classe qui permet de gérer les requêtes sur chaque table.
Prenom l'exemple de la table Joueur (généralisable aux autres). Dans le code PHP on retrouve une classe de nom Joueur mise dans le fichier Joueur.class.php . Dans cette classe on stockera toutes les informations relatives à un joueur. On trouve aussi une classe JoueurManager dans le fichier JoueurManager.class.php. Le manager est le fichier qui envoie les requêtes à MySQL et reçois les résultat, c'est en quelque sorte 'interface entre l'application web et la base de données. Cette séparation permet de bien cerner les tâches de chaque fichier et de rendre le code réutilisable modulaire et facile à comprendre. De plus le choix de mettre les requêtes dans des fichiers à part entière permet à toute personne disposant des sources de facilement visualisé toutes les requêtes implémentés. On peut ici définir une convention de nommage pour les classes.
– NomTable.class.php pour stocker les informations sur un élément de la base.
– NomTableManager.class.php pour gérer et exécuter les requêtes relatives à cette table.
Une fois les requêtes exécutées par le manger de la table adéquate ce dernier renvoie des objets du même type que le nom de la table. Ces objets sont ensuite formatés et présentés à l'utilisateur à l'aide de pages qui permettent de créer des vues pour l'utilisateur. Ces pages ne portent pas forcement le nom des tables car on doit afficher, par xemple, des notes avec des jeux. Donc une page du type noteVue.php n'est pas appropriée. Enfin vous pourrez consulter rapidement toutes les requêtes mises en oeuvre dans les fichier
L'interface PHP implémente aussi des contrôleurs qui permettent de vérifier la présence et la validité des données saisie par l'utilisateur. Ces contrôleur sont reconnaissable par leur nom , par exemple (jeuControl.php) il permettent aussi de contrôler si un utilisateur a le droit d'effectuer une action ou non (supprimer un joueur quand on est pas root)

3 Installation et utilisation

3.1 Mise en place du projet

L'installation est relativement facile. Il faut commencer par mettre en place la base de donnée. Il est préférable de créer une base dédié au project (nommé jeux-vidéos au mieux). Dans cette base, il faut exécuter le script de création des tables et remplir cette base avec les données du fichier donnes.sql. Ensuite il faut effectuer la préparation du script PHP. Il faut commencer par mettre les sources dans un serveur interprétant le PHP et modifier le fichier `PDOFactory.class.php`. Il faudra renseigner les champs nécessaire à la connexion à la base qui sont
– user : l'utilisateur de la base de donné
– mdp : mot de passe de user

– host : l'adresse du serveur de la base de donné
– dbname : le nom de la base de donnée utilisée

3.2 Utilisation

Le script est maintenant prêt à l'utilisation. Il ne reste plus qu'a ouvrir la page d'accueil. Le menu à gauche vous permet de naviguer entre les pages du site et de consulter le résultat de toutes les requêtes. La page d'accueil vous présente toutes les pages ainsi que les informations qu'elles exposent. Enfin le mot de passe de tous les utilisateur est *default*. L'utilisateur de pseudo *root* est le super-utilisateur qui peut faire toutes les modifications, les pseudos des autres joueurs sont disponible sur la page *Joueurs* et peuvent être utilisé pour tester la vue d'un utilisateur lambda.

4 Problèmes et Améliorations

4.1 Problèmes rencontrés

4.1.1 Incompatibilité Oracle/MySql

L'implémentation de la base de données en MySql nous a posé certains problèmes que nous avons rapidement réussi à surmonter. Nous avons essayé d'utiliser les scripts sur les machines de l'école tournant sous Oracle et la différence de syntaxe n'a pas permis de le faire, ce qui nous a contraint à supprimer certains mots clés tel `AUTO_INCREMENT` et à les remettre par la suite car ils étaient indispensables sous MySql pour garantir l'intégrité de la base.
Un autre problème rencontré est l'impossibilité d'utiliser un trigger `instead of` de la même manière que sur Oracle pour pouvoir, par exemple, interrompre l'insertion d'une note supérieure à 20. Toutes les vérifications de ce type n'ont pas pu être faite directement dans le SGBD et ont été effectué dans l'interface PHP.

4.1.2 Différences de version MySql

Nous avons remarqué la présence d'une différence entre version de MySql, une même requête répondait différemment sur *MySql 5.5.34* et *MySql 5.6.12*. Nous avons alors procédé à une vérification du bon fonctionnement des requêtes potentiellement problématiques sur les différentes versions qui étaient à notre disposition.

4.2 Améliorations possibles

Le projet rendu répond à toutes les requêtes demandé par la version avancé. Cependant il peut encore être amélioré. On pourrait ajouter un attribut groupe pour les joueur. Cet attribut permettra de gérer les droit de chaque utilisateur en fonction de son groupe (utilisateur, modérateur, administrateur...). D'autres améliorations sont envisageables et le sujet reste très ouvert.

Conclusion

En conclusion nous pouvons dire que le projet a été une grande occasion de manipuler une base de donnée depuis la modélisation à l'utilisation en passant par l'implémentation. Ceci constitue un excellent exercice à la fois théorique (pour la partie modélisation) et pratique (pour la partie implémentation). De plus le projet constitue un bon entraînement au travail d'équipe, notamment la manière de se répartir les tâches et de rester en contact pour avoir un livrable qui fonctionne, et le plus important c'est la réflexion groupé autour de la manière de modéliser la base où tous un chacun donne se idées.