Cours de Bases de données : Projet (seconde partie)

Antoine WEHENKEL Sébastien JORIS Thibaut THEATE

2 mai 2016

Question 1

Dans un premier temps, un script PHP a été écrit afin de générer la base de données et l'initialiser sur un serveur MySQL. Vous trouverez en annexe l'ensemble des requêtes SQL utilisées dans le script PHP.

Création de la base de données

Ce premier script SQL (voir annexe) permet de créer les différentes tables de la base de données. Il permet également de spécifier les multiples contraintes gouvernant cette base de données.

Lors de la création de la base de données, on a toutefois noté quelques observations :

- Un index a dû être rajouté sur l'attribut *id_ exemplaire* dans la table *Exemplaire*.
- Le type **date** était tout indiqué pour les attributs *date_ emprunt* et *date_ retour* mais n'a pas pu être utilisé car le format des dates du fichier *data.csv* n'est pas compatible avec ce type. En effet, les dates contenues dans le fichier *data.csv* sont du format JOUR-MOIS-ANNÉE alors que le format du type **date** est ANNÉE-MOIS-JOUR. Pour résoudre ce problème, on a utilisé le type **varchar(10)** tout en n'oubliant pas qu'on manipule des dates.

Initialisation de la base de données

Un script PHP permet ensuite d'initialiser la base de données avec les informations contenues dans le fichier *data.csv*. En effet, ce script lit le fichier *csv* et exécute les requêtes **INSERT INTO** pour chacune des tables de la base de données. Il détecte automatiquement le changement de table dans le fichier *csv*. Ainsi, si le fichier *data.csv* est modifié ultérieurement, l'initialisation de la base de données restera correcte.

Question 2

Dans un deuxième temps, des scripts PHP ont été écrits afin d'implémenter une interface web permettant d'accéder à la base de données créée et d'en manipuler les données.

Cette interface comprend un contrôle d'accès basé sur un utilisateur et un mot de passe. Ainsi, un script SQL a été écrit (cf *Annexe*) afin d'ajouter une nouvelle table (**users**) à la base de données, reprenant les utilisateurs et mots de passe associés. Une fois connecté, l'utilisateur est redirigé sur une page lui permettant de choisir entre 5 possibilités correspondant aux différentes questions ci-dessous.

Nous avons essayé d'optimiser les requêtes SQL en réduisant la taille des tables avant d'effectuer des joints (les opérations de sélection et projection sont toujours effectuées avant les opérations de jointure).

Question a

Cette page permet de, pour chaque table, sélectionner et afficher les tuples désirés en contraignant la valeur d'un ou plusieurs de ses champs. Veuillez observer que tous les tuples seront sélectionnés si le caractère * est utilisé.

Le script SQL utilisé pour implémenter cette question est fourni ci-dessous :

```
SELECT *
```

```
FROM table
WHERE restrictions;
```

Question b

Cette page permet d'ajouter un nouvel exemplaire à la bibliothèque. A cette fin, l'utilisateur devra renseigner le jeu-vidéo, la plateforme, le type (virtuel ou physique), et le cas échéant les émulateurs permettant de faire tourner l'exemplaire virtuel. Veuillez noter que l'utilisateur choisit ces derniers renseignements à partir de ceux déjà présents dans la base de données.

Le script SQL permettant d'exécuter ces opérations est fourni ci-dessous :

```
/* Si on ajoute un exemplaire physique */

INSERT INTO exemplaire
VALUES (id_jeu, id_exemplaire, id_plateforme);
INSERT INTO exemplaire_physique
VALUES (id_jeu, id_exemplaire, etat, emballage, livret);

/* Si on ajoute un exemplaire virtuel */

INSERT INTO exemplaire
VALUES (id_jeu, id_exemplaire, id_plateforme);
INSERT INTO exemplaire_virtuel
VALUES (id_jeu, id_exemplaire, taille);
INSERT INTO peut_emuler
VALUES(id_jeu, id_exemplaire, id_emulateur);
```

Question c

Cette page permet de retrouver, pour chaque style de jeu, le nombre d'exemplaires fonctionnels dans la base de données.

Le script SQL nécessaire pour réaliser cette opération est fourni ci-dessous. Voici une courte description de la requête :

- Sélection des exemplaires physiques dont l'état est supérieur à 1.
- En sélectionnant les *id_exemplaire* depuis la table *peut_emuler*, nous obtenons directement tous les exemplaires émulables. Ensuite, grâce à un joint avec *emulateur_fonctionne_sur*, nous obtenons tous les émulateurs fonctionnant sur des plateformes. Ce qui nous donne tous les exemplaires virtuels fonctionnels.
- L'union des deux dernières tables et le groupement par style nous donne le résultat souhaité.

```
SELECT style, COUNT(id_exemplaire) AS fonctionnel
FROM jeu_video
      NATURAL JOIN
       ((SELECT id_jeu, id_exemplaire
         FROM exemplaire_physique
         WHERE etat > 1)
```

```
        UNION

       (SELECT id_jeu, id_exemplaire
        FROM peut_emuler NATURAL JOIN (SELECT DISTINCT id_emulateur
                                       FROM emulateur_fonctionne_sur)AS T2)
        ) AS T1
GROUP BY style;
```

Question d

Cette page permet de trier les émulateurs par performance et d'afficher les résultats.

Le script SQL permettant d'implémenter cette opération est fourni ci-dessous. Voici une brève description de cette requête :

- On sélectionne dans un premier temps le nombre d'exemplaires réellement émulés par émulateur.
- On sélectionne ensuite les exemplaires et leurs plateformes correspondantes.
- En joignant cette dernière table à *emule*, en groupant par émulateur et en effectuant un **COUNT(id_exemplaire)**, on peut alors obtenir le nombre total d'exemplaires sensés fonctionner pour chaque émulateur.
- En calculant les ratios qui nous intéressent pour obtenir des performances, on peut alors trier nos émulateurs.

```
SELECT id_emulateur , nbreVirtuel/nbreVirtuel_Total AS performance
FROM
     (SELECT id_emulateur, COUNT(id_exemplaire) AS nbreVirtuel
      FROM peut_emuler
      GROUP BY id_emulateur) AS T1
      NATURAL JOIN
     (SELECT id_emulateur,COUNT(id_exemplaire) AS nbreVirtuel_Total
      FROM
          emule
         NATURAL JOIN
          (SELECT id_plateforme, id_exemplaire
           FROM
               exemplaire
              NATURAL JOIN
               (SELECT id_jeu, id_exemplaire
                FROM exemplaire_virtuel) AS T2) AS T2

GROUP BY id_emulateur) AS T2
ORDER BY performance DESC;
```

Question e

Cette dernière page crée une liste de recommandations pour un certain ami renseigné par l'utilisateur à partir de sa liste de prêts.

Le script SQL permettant de réaliser cette opération est fourni ci-dessous. Vous pouvez également trouver ci-dessous une courte description de la requête SQL :

- On sélectionne dans un premier temps les jeux-vidéos qui n'ont pas encore été prêtés à un certain ami.
- On sélectionne ensuite les jeux-vidéos dont le style est apprécié par cet ami (déjà joué).
- On sélectionne après cela les jeux-vidéos dont on possède un exemplaire physique utilisable sur une plateforme que possède cet ami (déjà joué).
- Finalement, on sélectionne les 5 premiers jeux-vidéos (triés par leur note) respectant les 3 contraintes énoncées ci-dessus.

```
SELECT id_jeu, style, note
FROM jeu_video
WHERE id_jeu NOT IN (SELECT id_jeu
                     FROM pret
                     WHERE id_ami = id_ami)
AND style IN (SELECT style
              FROM pret NATURAL JOIN jeu_video
              WHERE id_ami = id_ami)
AND id_jeu IN (SELECT id_jeu
               FROM exemplaire NATURAL JOIN exemplaire_physique
               WHERE id_plateforme IN (SELECT id_plateforme
                                       FROM pret NATURAL JOIN exemplaire
                                       WHERE id_ami = id_ami))
ORDER BY note DESC
LIMIT 5;
```

Implémentation générale de l'interface web

Afin de réaliser notre interface web, nous avons eu recours à plusieurs langages de programmation web :

- HTML : partie statique de l'interface web (qui n'interagit pas avec la base de données).
- CSS : partie esthétique de l'interface web.
- PHP : partie dynamique de l'interface web (qui interagit avec la base de données).
- JavaScript : dynamise un peu plus l'interface web.

Lien vers le site

Vous trouverez ci-dessous un lien vers une version opérationnelle de notre site et contenant uniquement les informations du fichier de données.

Lien : `http://www.student.montefiore.ulg.ac.be/~s131697/bdd-project`

Annexe

```
/* Scripts SQL permettant la creation de la base de donnees. */

Create table if not exists plateforme(
        id_plateforme int primary key,
        nom varchar(20) not null,
        version int not null,
        bits int not null
        )ENGINE=INNODB;

create table if not exists console(
        id_plateforme int primary key,
        Foreign key (id_plateforme) references plateforme(id_plateforme)
)ENGINE=INNODB;

create table if not exists systeme(
        id_plateforme int primary key,
        Foreign key (id_plateforme) references plateforme(id_plateforme)
)ENGINE=INNODB;

create table if not exists emulateur(
        id_emulateur int primary key,
        nom varchar(20) not null,
        version int not null
)ENGINE=INNODB;

create table if not exists ami(
        id_ami int primary key,
        nom varchar(20) not null,
        prenom varchar(20) not null,
        n_tel varchar(9) not null
)ENGINE=INNODB;

create table if not exists jeu_video(
        id_jeu int primary key,
        style varchar(20) not null,
        note int not null
)ENGINE=INNODB;

create table if not exists exemplaire(
        id_jeu int not null,
        id_exemplaire int not null,
        id_plateforme int not null,
        primary key(id_jeu, id_exemplaire),
        Foreign key (id_jeu) references jeu_video(id_jeu),
        Foreign key (id_plateforme) references plateforme(id_plateforme)
)ENGINE=INNODB;

ALTER TABLE `exemplaire` ADD KEY `id_exemplaire` (`id_exemplaire`);
```

```
create table if not exists exemplaire_physique(
        id_jeu int not null,
        id_exemplaire int not null,
        etat int not null,
        emballage boolean not null,
        livret boolean not null,
        primary key(id_jeu, id_exemplaire),
        Foreign key (id_jeu) references jeu_video(id_jeu)
)ENGINE=INNODB;

create table if not exists exemplaire_virtuel(
        id_jeu int not null,
        id_exemplaire int not null,
        taille int not null,
        primary key(id_jeu, id_exemplaire),
        Foreign key (id_jeu) references jeu_video(id_jeu)
)ENGINE=INNODB;

create table if not exists plateforme_du_jeu(
        id_jeu int not null,
        id_plateforme int not null,
        primary key(id_jeu, id_plateforme),
        Foreign key (id_plateforme) references plateforme(id_plateforme),
        Foreign key (id_jeu) references jeu_video(id_jeu)
)ENGINE=INNODB;

create table if not exists peut_emuler(
        id_jeu int not null,
        id_exemplaire int not null,
        id_emulateur int not null,
        primary key(id_jeu, id_exemplaire, id_emulateur),
        Foreign key (id_jeu) references jeu_video(id_jeu),
        Foreign key (id_exemplaire) references exemplaire(id_exemplaire),
        Foreign key (id_emulateur) references emulateur(id_emulateur)
)ENGINE=INNODB;

create table if not exists emulateur_fonctionne_sur(
        id_plateforme int not null,
        id_emulateur int not null,
        primary key(id_emulateur, id_plateforme),
        Foreign key (id_plateforme) references plateforme(id_plateforme),
        Foreign key (id_emulateur) references emulateur(id_emulateur)
)ENGINE=INNODB;

create table if not exists emule(
        id_emulateur int not null,
        id_plateforme int not null,
        primary key(id_emulateur, id_plateforme),
        Foreign key (id_plateforme) references plateforme(id_plateforme),
        Foreign key (id_emulateur) references emulateur(id_emulateur)
)ENGINE=INNODB;
```

```
create table if not exists pret(
        id_ami int not null,
        id_jeu int not null,
        id_exemplaire int not null,
        date_emprunt varchar(10) not null,
        date_retour varchar(10) null,
        primary key(id_jeu, id_exemplaire, date_emprunt),
        Foreign key (id_jeu) references jeu_video(id_jeu),
        Foreign key (id_exemplaire) references exemplaire(id_exemplaire)
)ENGINE=INNODB;

create table if not exists users(
        id varchar(20) NOT NULL,
        password varchar(20) NOT NULL
)ENGINE=INNODB;


/* Scripts SQL permettant d'initialiser la base de donnees. */

INSERT INTO users(id, password) VALUES ('antoine', 'password');
INSERT INTO users(id, password) VALUES ('group8', 'IR5ovtPs');

/* En ce qui concerne l'importation du fichier data.csv: voir rapport. */
```