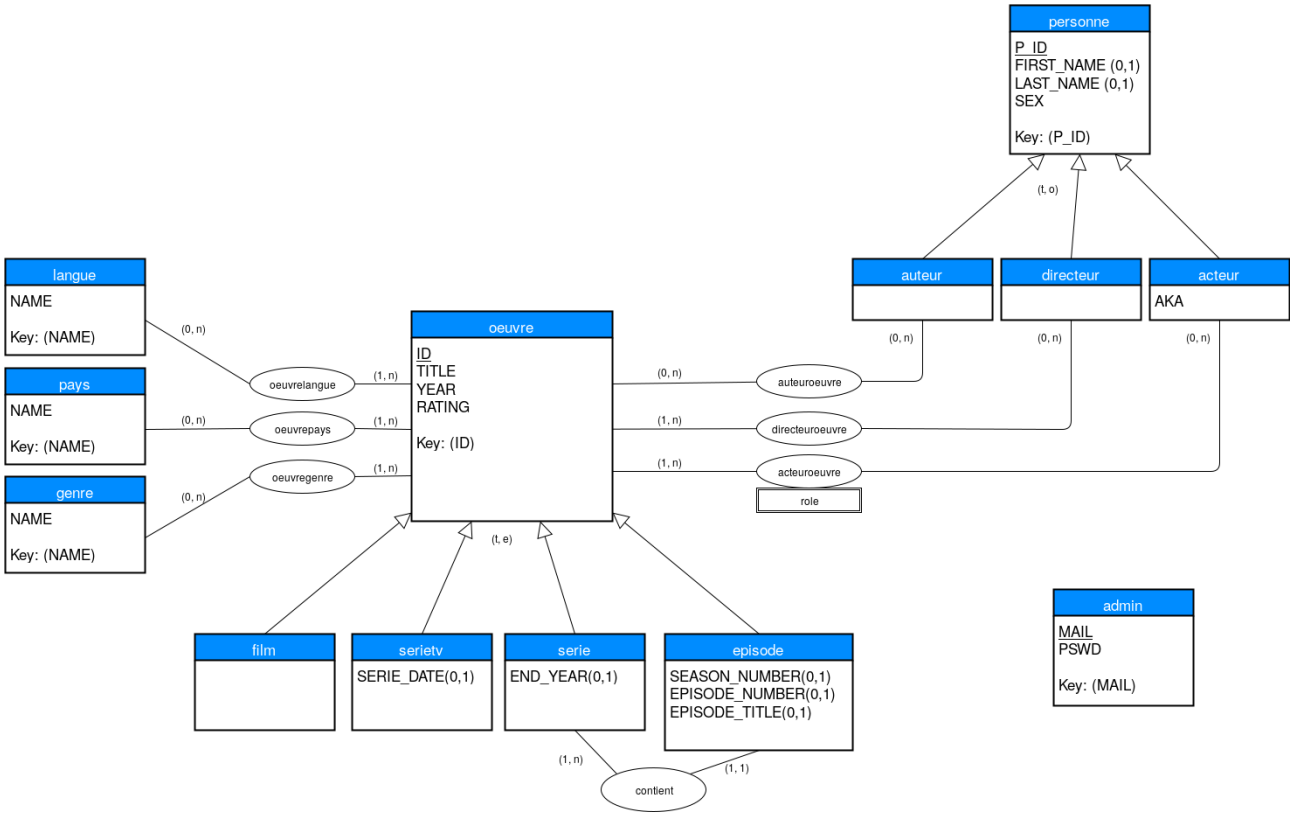
INFO-H303 - Rapport du Projet

CHOUJAÂ Wassil - *408773* *wchoujaa@ulb.ac.be*
COSTE-ALEXEY Rumuri - *414706* *crumuri@ulb.ac.be*
NOGUEIRA Pedro - *414153* *pnogueir@ulb.ac.be*

18 mai 2017

Diagramme Entité-Association



Traduction Relationel

acteur (ID, AKA)
ID reference personne.P_ID

acteuroeuvre (OID, ID)
OID reference oeuvre.ID
ID reference personne.P_ID

admin (MAIL, PSWD)

auteur (ID)
ID reference personne.P_ID

auteuroeuvre (OID, ID, ROLE)
OID reference oeuvre.ID
ID reference personne.P_ID

directeur (ID)
ID reference personne.P_ID

directeuroeuvre (OID, ID)
OID reference oeuvre.ID
ID reference personne.P_ID

episode (OID, SERIE_ID, SEASON_NUMBER, EPISODE_NUMBER, EPISODE_TITLE)
OID reference oeuvre.ID
SERIE_ID reference serie.OID

film (OID)
OID reference oeuvre.ID

genre (NAME)

langue (NAME)

oeuvre (ID, TITLE, YEAR, RATING)

oeuvregenre (NAME, ID)
NAME reference genre.NAME
ID reference oeuvre.ID

oeuvrelangue (NAME, ID)
NAME reference langue.NAME
ID reference oeuvre.ID

oeuvrepays (NAME, ID)
NAME reference langue.NAME
ID reference oeuvre.ID

pays (NAME)

personne (P_ID, FIRST_NAME, LAST_NAME, SEX)

serie (OID, END_YEAR)
OID reference oeuvre.ID

serietv (OID, SERIE_DATE)

Contraintes d'integrité

— L'année de fin d'une série doit toujours être postérieur à la date de début.
— Deux épisodes distincts d'une même saison d'une même série doivent obligatoirement avoir un numéro d'épisode différent.
— L'année de fin d'une série est optionel car la série peut ne pas être achevée.
— L'année de création d'un épisode doit être supérieur ou égale à l'année de création de la série qu'il compose et inférieur ou égal à son année de fin.
— Une oeuvre peut avoir plusieurs lieux de tournage, plusieurs langues, plusieurs genres et plusieurs acteurs, directeurs et auteurs.
— Il est possible que le lieu de tournage (PAYS) d'un épisode ne se trouve pas dans les lieux de tounage de la série, de même que pour sa langue et son genre.
— Il est aussi possible qu'une série n'ait pas tous les acteurs qui ont participé dans chaque épisode (IMDB n'est pas 100% correcte).

Hypothèses et Justifications

Pour le choix des clefs, nous nous sommes adaptés aux contraintes physiques de la base de données d'*IMDB*. Du fait de la taille de la base de données, il nous a fallu mettre en place un système d'*ID* pour faciliter la recherche.

Dans l'énoncé il est précisé que chaque acteur a un nom, prénom et genre. Rien n'est dit à propos des directeurs ni des auteurs. On a supposé qu'ils en ont un aussi. C'est pourquoi nous avons généralisé ces données-là par une nouvelle entité : personne, qui n'a qu'une utilité symbolique.

Languages utilisées

— **Python** : Ulitisée pour le *DLL* pour créer la base de données et lors du parsing afin de convertir les fichiers *.list* dans une base de données *mySQL*.
— **SQL** : Pour manipuler la base de données à travers des requêtes.
— **HTML** et **CSS** : Pour faire la page web, *HTML* pour le corps de la page et *CSS* pour la rendre plus agréable a la vue.
— **PHP** : Il s'exécute du côté du serveur pour accéder à la base de données et renvoyer les résultats ao client.
— **JavaScript** : Utilisé pour le *Quizz*, entre autres choses. Executé du côté client, rend la page dynamique en permettant des altérations dans le contenu.
— **AJAX** : C'est un acronyme pour *Asynchronous JavaScript And XML*. Il s'exécute aussi du côté client et permet de saisir des données en parallele sans avoir la nécessité de recharger la page.

Méthode d'extraction des données

Pour réaliser l'extraction des données, un parsing a été réalisé.
Le parsing a été réalisé en *Python* pour faire l'extraction de l'information des fichiers *.list*.
Le même *script* contient l'injection des données dans une base de données du type *mySQL*.
Afin de garantir l'intégrité des clefs le moteur *mySQL INODB* a été activé lors du parsing.

Requêtes SQL

R1 :

```
SELECT P.FIRST_NAME, P.LAST_NAME
FROM personne P
WHERE P.P_ID IN (
    SELECT ACTEURID
    FROM (
        SELECT AO.ID as ACTEURID
        FROM acteuroeuvre AO , oeuvre O
        WHERE AO.OID = O.ID AND O.YEAR BETWEEN 2003 AND 2007
        GROUP BY AO.ID , O.YEAR) as subsubquery
    GROUP BY ACTEURID
    HAVING COUNT(*) = 5)
```

R2 :

```
SELECT distinct P.FIRST_NAME, P.LAST_NAME
FROM personne P, (
    SELECT auteuroeuvre.id , oeuvre.year as In_Year,
     count(*) as NumberOfFilms
```

```
    FROM auteuroeuvre, film, oeuvre
    WHERE film.oid = auteuroeuvre.oid  and oeuvre.id = film.oid
    GROUP BY auteuroeuvre.id, oeuvre.year
    HAVING NumberOfFilms > 1 ) as res1
WHERE P.P_ID = res1.id
```

R3 :

```
set sql_mode="";

# Creates table with all the filmes and actors (X | FILM)
CREATE TEMPORARY TABLE IF NOT EXISTS tmp1 (
    SELECT acteuroeuvre.id as X, acteuroeuvre.oid as FILM
    FROM acteuroeuvre, film
    WHERE acteuroeuvre.oid = film.oid
    ORDER BY acteuroeuvre.oid);

# Copy from the 1st table (Y | FILM)
CREATE TEMPORARY TABLE IF NOT EXISTS tmp2 (
    SELECT X as Y , FILM
    FROM tmp1);

# Makes a natural join with the 1st two tmp tables (based on the film)
#(FILM1 | X | Y)
CREATE TEMPORARY TABLE IF NOT EXISTS tmp3(
    SELECT FILM as FILM1, X, Y
    FROM tmp1
    NATURAL JOIN tmp2
    WHERE X != Y);

# Copy from the 3rd table (FILM2 | Y | Z)
CREATE TEMPORARY TABLE IF NOT EXISTS tmp4(
    SELECT FILM1 as FILM2, X as Y , Y as Z
    FROM tmp3);

# Joins tmp3 and tmp4 (based on Y)
CREATE TEMPORARY TABLE IF NOT EXISTS RESULT(
    SELECT X , FILM1 ,Y ,FILM2 , Z
    FROM tmp3
    NATURAL JOIN tmp4
    WHERE  FILM1 != FILM2 AND X!=Z);
```

```
# We no longer need these two tables
DROP TEMPORARY TABLE tmp3;
DROP TEMPORARY TABLE tmp4;

# Makes sure that there is no path of distance 1
SELECT P1.FIRST_NAME, P1.LAST_NAME, P2.FIRST_NAME, P2.LAST_NAME
FROM personne P1, personne P2,(
    SELECT R.*
    FROM RESULT R
    WHERE NOT EXISTS(
        SELECT *
        FROM tmp1, tmp2
        WHERE tmp1.X != tmp2.Y and R.X = tmp1.X and
            R.Z = tmp2.Y and tmp1.FILM = tmp2.FILM)) as R
WHERE P1.P_ID = R.X AND P2.P_ID = R.Z
ORDER BY P1.FIRST_NAME, P1.LAST_NAME, P2.FIRST_NAME, P2.LAST_NAME;

# Destroys all remaining temporary tables
DROP TEMPORARY TABLE tmp1;
DROP TEMPORARY TABLE tmp2;
DROP TEMPORARY TABLE RESULT;
```

R4 :

```
SET SQL_MODE ="" ;

SELECT E.*
FROM episode E
WHERE E.OID NOT IN (
    SELECT AO.OID
    FROM  acteuroeuvre AO, episode E,personne P
    WHERE E.OID = AO.OID AND AO.ID = P.P_ID
    GROUP BY P.SEX , AO.OID
    HAVING P.SEX = 'M') # episode ayant au moin un acteur masculin
```

R5 :

```
SELECT p.first_name, p.last_name, nb_serie_joue
FROM personne p, (
    SELECT AO.ID as acteurID , Count(*) as nb_serie_joue
    FROM acteuroeuvre AO ,serie S
    WHERE AO.OID = S.OID
    GROUP BY AO.ID) as res
```

```
WHERE p.p_id = acteurID
ORDER BY res.nb_serie_joue DESC LIMIT 10
```

R6 :

```
# Get ID, title, rating and begin and end year
CREATE TEMPORARY TABLE IF NOT EXISTS tmp_serie(
    SELECT O.ID, O.TITLE, O.RATING, O.YEAR,
        COALESCE(S.END_YEAR, '2010') AS END_YEAR
    FROM serie S, oeuvre O
    WHERE S.OID = O.ID);

# Calculates the average of ratings
CREATE TEMPORARY TABLE IF NOT EXISTS tmp_AVG(
    SELECT AVG(tmp.RATING) as R
    FROM tmp_serie tmp);

# Takes only the series above average and adds the total of episodes
CREATE TEMPORARY TABLE IF NOT EXISTS tmp_S_TE(
    SELECT tmp_S.*,
        (SELECT COUNT(*)                                    # Total of episodes
        FROM episode E
        WHERE tmp_S.ID = E.SERIE_ID) as Total_Episodes,
        (SELECT REPLACE(MAX(E.SEASON_NUMBER), '-1', '1')    # Number of seasons
        FROM episode E
        WHERE tmp_S.ID = E.SERIE_ID) as NB_Seasons,
        (SELECT DISTINCT COUNT(AO.ID)                       # Number of actors
        FROM acteuroeuvre AO
        WHERE tmp_S.ID = AO.OID) as NB_Actors
    FROM tmp_serie tmp_S, tmp_AVG
    WHERE tmp_S.RATING > tmp_AVG.R);

# These temporary tables are no longer needed.
DROP TEMPORARY TABLE tmp_serie;
DROP TEMPORARY TABLE tmp_AVG;

# Takes the third table, calculates the averages and return the asked columns
SELECT A.TITLE, A.Total_Episodes,
    CAST(A.Total_Episodes/(A.END_YEAR-A.YEAR+1) AS DECIMAL(10,2))
    AS AVG_Episodes_Year,
    CAST(A.NB_Actors/(A.NB_Seasons) AS DECIMAL(10,2)) AS AVG_Actors_Season
FROM tmp_S_TE A
```

```
ORDER BY A.TITLE;

# Destroy temporary last table
DROP TEMPORARY TABLE tmp_S_TE;
```

Algèbre Relationel

R1 :

$$R_A \leftarrow oeuvre \bowtie_{ID=OID} \alpha_{ID:acteur\_ID}(acteuroeuvre)$$
$$R_B \leftarrow \sigma_{2003 \leq YEAR \leq 2007}(R_A)$$
$$R_C \leftarrow \pi_{acteur\_ID,YEAR}(R_B)$$
$$R_D \leftarrow \pi_{YEAR}(R_C)$$
$$R \leftarrow R_C \div R_C$$
$$Result \leftarrow \pi_{First\_Name,Last\_Name}(Personne *_{P\_ID=acteur\_ID} R)$$

R2 :

$$R_A \leftarrow \pi_{OID,YEAR}(\alpha_{ID:OID}(oeuvre))$$
$$R_B \leftarrow R_A *_{OID=OID} acteuroeuvre$$
$$R_C \leftarrow R_B *_{ID=ID} \alpha_{YEAR:YEAR2,OID:OID2}$$
$$R_D \leftarrow \pi_{ID}(\sigma_{YEAR=YEAR2 \wedge OID \neq OID2}(R_C))$$
$$Result \leftarrow \pi_{FIST\_NAME,LAST\_NAME}(Personne \bowtie_{P\_ID=ID} R_D)$$

R3 :

$$R_A \leftarrow acteuroeuvre *_{OID=OID} Film$$
$$R_B \leftarrow \pi_{ID,OID}(R_A)$$
$$R_C \leftarrow \alpha_{ID:X}(R_B) *_{OID=OID} \alpha_{ID:Y}(R_B)$$
$$R_D \leftarrow \alpha_{OID:Film1}(R_c) *_{Y=Y \wedge Film1 \neq Film2} \alpha_{OID:Film2 \wedge X:Y \wedge Y:Y2}(R_C)$$
$$R_E \leftarrow \pi_{X,Y2}(R_D)$$
$$R_F \leftarrow \alpha_{FIRST\_NAME:NAME\_X,LAST\_NAME:NAME2\_X}(Personne \bowtie_{X=P\_ID} R_E)$$
$$R_G \leftarrow \alpha_{FIRST\_NAME:NAME\_Y,LAST\_NAME:NAME2\_Y}(Personne \bowtie_{Y=P\_ID} R_F)$$
$$Result \leftarrow \pi_{NAME\_X,NAME2\_X,NAME\_Y,NAME2\_Y}(R_G)$$

R4 :

$$R_A \leftarrow \pi_{ID,OID,Serie\_ID(episode * acteuroeuvre)}$$
$$R_B \leftarrow R_A *_{ID=P\_ID} \pi_{ID,SEX}(personne)$$
$$R_C \leftarrow R_B \div \pi_{Sex}(\sigma_{sex=" M"}(personne))$$
$$R_D \leftarrow \pi_{OID}(R_A) - \pi_{OID}(R_C)$$
$$Result \leftarrow \pi_{Title,episode\_title,season,episode}(oeurve \bowtie_{serie\_ID=OID} R_E * episode)$$

R5 :

$$R_A \leftarrow \pi_{OID,ID}(acteuroeuvre *_{OID=OID} serie)$$
$$R_B \leftarrow ID\ F_{count(*)}(R_A)$$
$$R_C \leftarrow \pi_{ID,count,count2}(R_B \bowtie_{count \neq count2} \alpha_{count:count2}(R_B))$$
$$R \leftarrow \pi_{ID}(R_A) - \pi_{ID}(\sigma_{count<count2} R_C)$$
$$Result \leftarrow \pi_{First\_Name,Last\_Name}(personne \bowtie_{P\_ID=ID} R)$$

R6 :

$$R_A \leftarrow \pi_{OID,RATING}(serie \bowtie_{OID=ID} oeuvre)$$
$$AVERAGE \leftarrow F_{AVG(RATING)}(R_A)$$
$$R_B \leftarrow \sigma_{RATING>AVERAGE}(R_A * AVERAGE)$$
$$R \leftarrow OID, RATING\ F_{count(*)}(R_B \bowtie_{OID=SERIE\_ID} episode)$$
$$Result(Serie, Rating, Episode\_totaux) \leftarrow R$$

Calcul Relationel Tuple

R1 :

$$\{P.FIRST\_NAME, P.LAST\_NAME \mid personne(P) \forall$$
$$P.P\_ID = AO.ID(acteuroeuvre(AO) \wedge \exists O(oeuvre(O) :$$
$$O_i.YEAR \neq O_j.YEAR \wedge 2003 \leq O_i, O_j \leq 2007) \wedge 1 \leq i, j \leq 5)\}$$

R2 :

$$\{P.FIRST\_NAME, P.LAST\_NAME \mid personne(P) \forall$$
$$P.P\_ID = AO_i.ID(auteuroeuvre(AO_i, AO_j) \wedge$$
$$\exists Oeuvre(O_i, O_j) \wedge film(F) :$$
$$(AO_i.ID = AO_j.ID \wedge AO_i.OID \neq AO_j.OID \wedge$$
$$AO.OID \in F.OID \wedge AO_i.OID = O_i.ID \wedge AO_j.OID = O_j.ID \wedge O_i.YEAR = O_j.YEAR))\}$$

R3 :

$$\{P_i.FIRST\_NAME, P_i.LAST\_NAME, P_j.FIRST\_NAME, P_j.LAST\_NAME \mid$$
$$personne(P_i, P_j) \forall P_i.P\_ID = AO_i.ID \wedge P_j.P\_ID = AO_j.ID$$
$$(acteuroeuvre(AO_i, AO_j, AO_k) \wedge AO_i \neq AO_j \neq AO_k \wedge$$
$$\exists AO_i \cap AO_k \wedge \exists AO_i \cap AO_k \wedge \nexists AO_i \cap AO_j\ )\}$$

R4 :

$$\{E.OID, E.SERIE\_ID, E.SEASON\_NUMBER, E.EPISODE\_NUMBER,$$
$$E.EPISODE\_TITLE \mid Episode(E) \wedge$$
$$\nexists AO(acteuroeuvre(AO) \wedge \exists P(personne(P)$$
$$E.ID = AO.OID \wedge AO.ID = P.P\_ID \wedge P.SEX = "M"))\}$$

R5 :

$$\{P.FIRST\_NAME, P.LAST\_NAME \mid personne(P) \forall$$
$$P.P\_ID = A_i.ID(auteur(A) \wedge 0 \leq i \leq 9 \wedge$$
$$\nexists B(acteur(B) \wedge (\#acteuroeuvre.ID = B.ID) > (\#acteuroeuvre.ID = B.ID))\}$$

R6 :

$$\{S, \#E, \frac{\#E}{\#OE.Year}, \frac{\#AO.ID}{\#E.Saison} \mid serie(E) \wedge$$
$$\exists O(oeuvre(O) \wedge O.ID = S.OID \wedge$$
$$O.RATING > \left(\frac{\sum oeuvre.RATING}{\#serie} \wedge oeuvre.ID \in serie.OID\right)) \wedge$$
$$\exists E(episode(E) \wedge E.SERIE\_ID = S.OID) \wedge$$
$$\exists OE(oeuvre(OE) \wedge OE.ID = E.OID \wedge) \wedge$$
$$\exists AO(auteuroeuvre(AO) \wedge AO.OID = E.OID)\}$$

Explications et Justifications

— **Requête 5** : On a choisi d'afficher uniquement le top 10 d'acteurs qui ont joué dans le plus de séries.

— **Requête 6** : Quand on n'a pas de date de fin d'une série, ça veut dire que la série n'a pas encore terminé et comme notre base de données va jusqu'à 2010, en prend 2010 comme année de fin, afin de pouvoir calculer la moyenne des épisodes par an.

— **Requête 6** : De même, pour calculer le nombre d'acteurs moyen par saison, si on ne sait pas le nombre de saisons, on prend 1 comme valeur (une saison).

Modifications et déléctions dans la base de données :

Pour garder la cohérence dans la base de données, quand on doit effacer un acteur, un auteur ou un directeur, on efface aussi l'acteuroeuvre, auteuroeuvre et directeuroeuvre respectivement.

Quand on doit effacer un film, une série, un épisode ou une serieTV on efface aussi tout ce qui est en lien avec lui, ce qui veut dire : les acteuroeuvres, les auteuroeuvres, les directeuroeuvres, les

oeuvrelangue, les oeuvrepays, les oeuvregenre et l'oeuvre. Quand c'est une série, on fait en plus la même chose pour chacun de ses épisodes.