« We must do what nobody doing for you… »

Sujet : Structure de base de données & Algorithme de suggestion

Auteurs : Valentin Duhamel, Clément Delain, Valentin Lecorps, Samuel Lioult, André Douglas Djomgoue Katchieu

Date : 26/06/15

Propriétaire : Société 4 Highway

Table des matières

1. Structure de base de données

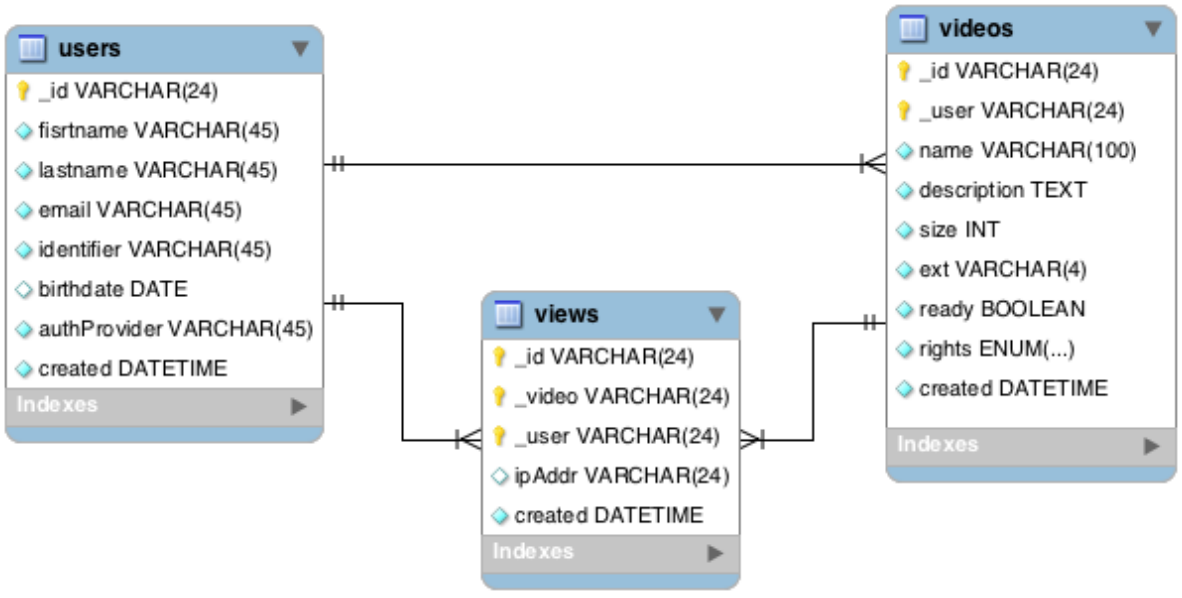
A. Présentation de mongoDB

MongoDB est une base de données orientée document répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Elle est utilisée notamment chez Foursquare, SourceForge et Bit.ly.

Mongo stocke les documents au format BSON (JSON binaire) et fournit un shell javascript pour accéder aux données et faire les opérations d'administration.

La notion de base orientée document signifie que les objets stockés sont représentés sous la forme d'un document BSON permettant facilement de se mapper sur les objets manipulés dans les programmes.

B. Représentation UML de la base de données



2. Algorithme de suggestion

A. Présentation de l'algorithme

Notre algorithme de suggestion est conçu pour retourner une liste de vidéos correspondant aux goûts de l'utilisateur connecté. Cela est possible grâce à l'enregistrement des vidéos visionnées pour chaque utilisateur. L'algorithme compare les différentes vidéos visionnées avec les autres vidéos afin de retourner les vidéos similaires.

B. Fonctionnement de l'algorithme

L'algorithme se décompose en plusieurs étapes :

- Récupération des vidéos visionnées par l'utilisateur :

```
models.View.find({_user: req.user._id})
.limit(10)
.populate("_video")
.exec(function(err, views){

});
```

- Décomposition du titre des vidéos pour en retourner un tableau de mots-clés :

```
var keywords = [];
for(var i=0; i<views.length; i++){
    var currentKeywords = views[i]._video.name.replace(/\_/g,' ').replace(/\./g,' ').split(" ");
    for(var y=0; y<currentKeywords.length; y++){
        keywords.push(currentKeywords[y]);
    }
}
keywords = modules._.uniq(keywords);
for(var i=0; i<keywords.length; i++){
    if(keywords[i].length <= 1){
        keywords.splice(i, 1);
        i--;
    }
}
```

- Récupération des vidéos ayant dans leur titre un ou plusieurs des mots-clés :

```
var suggestedVideos = [];
modules.async.each(keywords, function (keyword, callback){
    var regExSuggest = new RegExp(keyword, 'i');
    models.Video.find({rights: "public", name: { $regex: regExSuggest } })
    .where("archived").ne(true)
    .where("ready").equals(true)
    .populate("_user", "-identifier -__v")
    .select("-__v -archived")
    .lean()
    .exec(function(err, videos){
        if(videos){
            for(var i=0; i<videos.length; i++){
                suggestedVideos.push(videos[i]);
            }
        }
        callback();
    });
}
```

- Tri du tableau de vidéos pour retourner que les vidéos uniques et retirer les doublons :

```
var flags = [], uniqSuggestedVideos = [], l = suggestedVideos.length, i;
for(i=0; i<l; i++){
    if(flags[suggestedVideos[i]._id]) continue;
    flags[suggestedVideos[i]._id] = true;
    uniqSuggestedVideos.push(suggestedVideos[i]);
}
suggestedVideos = uniqSuggestedVideos;

if(!uniqSuggestedVideos){
    return res.json({"success": true, "data": []});
}
suggestedVideos = uniqSuggestedVideos;
```

- Tri du tableau pour retirer les vidéos déjà visionnées :

```
var videosViewed = [];
for(var i=0; i<views.length; i++){
    videosViewed.push(String(views[i]._video._id));
}
var fullSuggestedVideos = suggestedVideos;
for(var i=0; i<fullSuggestedVideos.length; i++){
    if(modules._.contains(videosViewed, String(fullSuggestedVideos[i]._id))){
        suggestedVideos.splice(i, 1);
        i--;
    }
}
```

- Réorganisation du tableau par ordre décroissant du nombre de vues par vidéos avant de renvoyer le tableau à l'utilisateur :

```
1  suggestedVideos = modules._.sortBy(suggestedVideos, 'views');
2  suggestedVideos = suggestedVideos.reverse();
3  res.json({"success": true, "data": suggestedVideos});
```