PROJET BASE DE DONNEES PARTIE 1 & 2

Etudiant : AZZOUG AGHILAS

Les choses réalisées	Les choses nos réalisées
✓ Schéma ✓ Première requête ✓ Deuxième requête ✓ Troisième requête ✓ 4eme requête (mais pas certain du résultat) ✓ Toutes les requêtes de partie 2 sauf une seule ✓ 1- 2-3 de fonctions et procédures ✓ 1-3-4 de contraintes	○ La requête 5 de la partie Une pas réalisée

Table des matières

Rapport du Projet Base de données 2 ------ Site Web (plateforme lecture vidéo) ----- Aghilas AZZOUG

Etudiant : Aghilas AZZOUG groupe 1

Mon rendu se compose de 5 fichiers ;(Tests, requête, tables, drop) en sql, rapport.

1 PARTIE 1 :

1.1 Objectif du Projet :

Mise en place d'une base de données pour un site internet permettant aux utilisateurs du site internet de regarder des émission (chaines audiovisuelles) et liker les épisodes ainsi d'avoir un compte pour regarder l'historique du visionnage et permettre de revoir le contenu des vidéos archivées qui ne sont plus disponibles sur Replay.

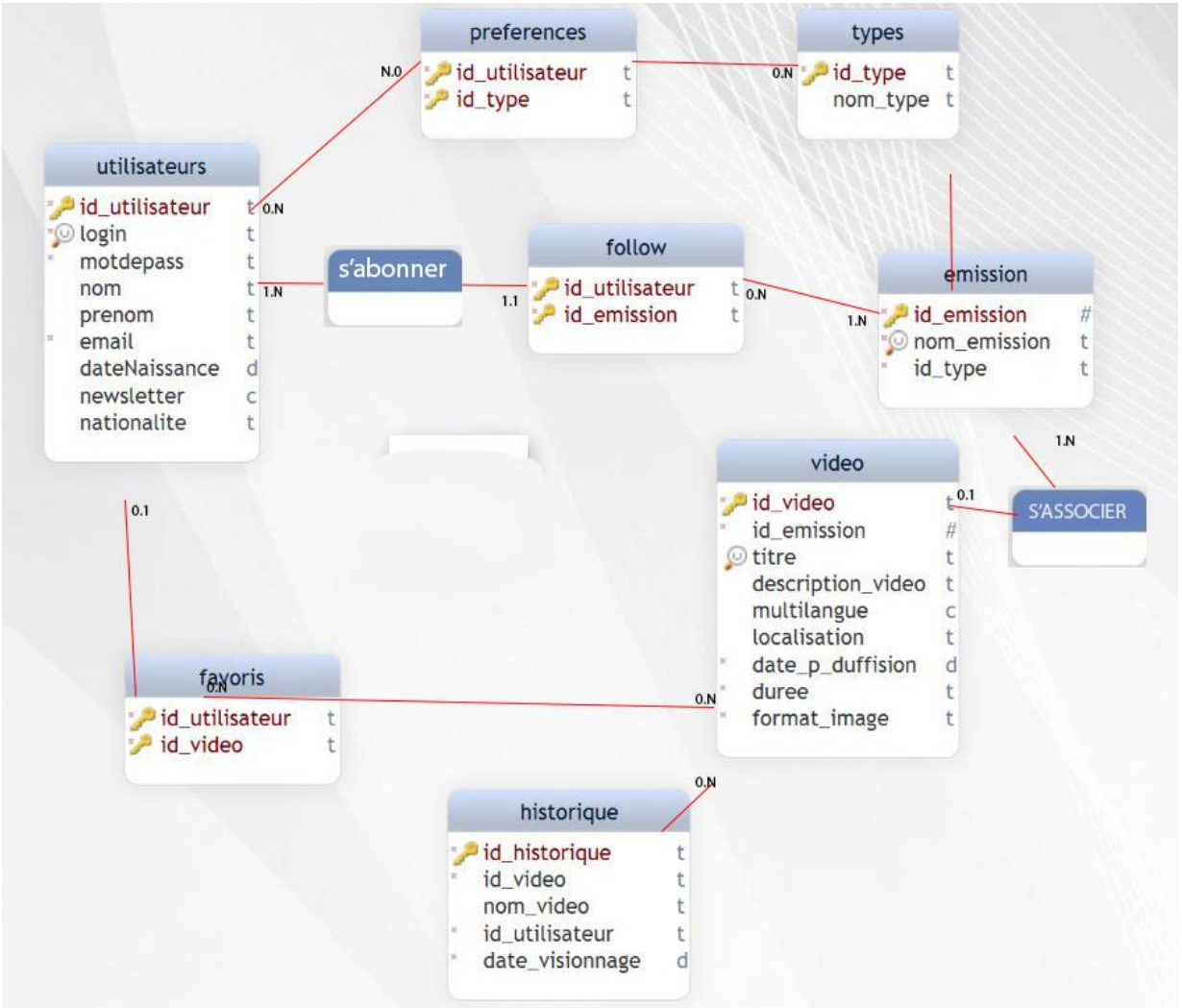
1.2 Procédures et méthodologie :

Étant donné que cette base de données sera utilisée pour une plateforme web (website), donc logiquement les traitements et les requêtes seront faits par le User (utilisateur du site) et d'autres par le gestionnaire du site (coté Back-End), mais pour faciliter et réduire le travail à ce dernier, je ferai en sorte de mettre plusieurs options et requêtes possibles pour le user.

1.3 Entités :

D'après la demande des développeurs Back-End(énoncées dans le sujet), j'ai décidé de mettre en place ces entités, capables à répondre aux besoins cités dans l'énoncé.

1.4 Schéma :



1.5 Premier modèle logique de la table :

Ce projet a été séparé en deux parties. La première fut simplement la conceptualisation de la table. Cela nous à mener à cette première modélisation. On avait ici un modèle sans redondance et sans incohérence.

1.6 Mes tables :

- TABLE utilisateurs;
- TABLE emission;
- TABLE video;
- TABLE favoris;
- TABLE types;
- TABLE follow;
- TABLE preferences;
- TABLE historique;
- SEQUENCE seq_utilisateurs;
- SEQUENCE seq_emission;
- SEQUENCE seq_video;
- SEQUENCE seq_type;

1.6.1 Utilisateurs :

Cette table permet de voir les utilisateurs qui sont inscrits sur le site :

Pour la newsletter j'ai mis un booléen, soit 1 si c'est ok soit c'est 0 (par default), je pourrai rajouter la contrainte de date qui ne doit pas avoir moins de 1ç années en arrière par rapport a la date d'aujourd'hui ou du moins ne pas la mettre antérieur à la date d'aujourd'hui. Mais cela n'a pas été demandé, je pourrais toujours la rajouter dans la deuxième partie.

```
-- Création de table "utlisateurs". Table qui permet à garder les utilisateurs
-- et ses données. Présence de newsletter ; pour determiner si un user accepte de recevoir
-- des notifications ou pas (news)
CREATE TABLE utilisateurs (#id_utilisateur,login,motdepasse,nom,prenom,dateNaissance,newsletter
,nationalite)
```

1.6.2 Emission :

```
-- Création de table "emission". Table qui permet de connaitre des informations sur le contenu
-- des émissions et catégories ...
create table emission (#id_emission, non_emission, id_type)
```

1.6.3 Types :

Je n'ai besoin que de ces deux attributs

```
-- Création de table "types". Table qui permet à garder les types/catégories des
-- émissions.
CREATE TABLE types (#id_type,nom_type)
```

1.6.4 Video :

Multilangue c'est un booléen soit 1 si c'est disponible soit c'est 0 par default , pour localisation c'est le pays d'origine, format image : dans ma base de donnée proposée j'ai mis 2K,HD,MP3

```
-- Création de table "video". Table qui permet de voir les informations sur la video.
CREATE TABLE video (#id_video,
#id_emission,titre,description_video,multilangue,localisation,anne_p_diffusion,duree,format_ima
ge)
```

1.6.5 Historique :

Je voulais au tout début mettre la table historique contenant la table des vidéos favorites, mais comme l'énoncé indique que les vidéos peuvent ne pas encore être vues, dans j'ai dû les séparer les deux tables et ça marche beaucoup mieux .

Id_video et id_utilisateur sont les clés estrangères, et me serviront pour mes requêtes.

le d'historique permet d'avoir un nombre répétitif de fois de visualisation de la vidéo par le même utilisateur

```
-- Mise en place la table Archives videos vu la nécessité de vouloir récuperer les videos vues
 qui peuvent être supprimées dans replay
CREATE TABLE historique(#id_historique,id_video,nom_video,id_utilisation,date_visionnage)
```

1.6.6 Favoris :

J'utilise les deux clés étrangères et qui seront mes clés composées. une liste des choix des vidéos a voir prochainement.

```
-- Mise en place la table Favoris pour pouvoir avoir une listes des vidéos likées (aimées)
CREATE TABLE favoris (#id_utilisateur,#id_video)
```

1.6.7 Préférences :

Permets l'user de garder ses préférences comme catégories, et le site pourrait s'en servir pour lui servir du contenu pertinent comme suggestion

```
-- Mise en place la table préférences pour
que le user puisse indiquer ses préférences de catégories(types)
CREATE TABLE preferences(#id_utilisateur, #it_type)
```

1.6.8 Follow:

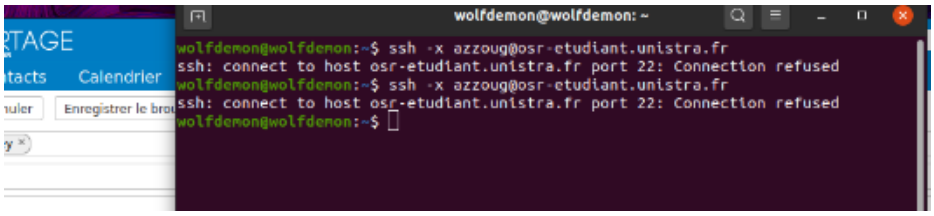
Le user peut s'abonner a plusieurs émissions.

```
-- Création de table "follow". Table qui permet de voir les utilisateurs
--abonnés à cette émission et permet aussi de voir les abonnements de l'user
CREATE TABLE follow (#id_utilisateur, #id_emission)
```

1.7 Les difficultés rencontrées pour la réalisation du projet et comment j'ai pu les résoudre :

Au tout début je ne savais pas trop par où commencer, je suis resté Deux semaines à attendre à installer oracles, pour des soucis du package JAVA 11 qui est demandé par le logiciel, je n'ai pas pu arriver à l'installer, puis le temps que je m'adapte au WORKBENCH, j'ai du après revenir à la base, en me servant de mon terminal et me connectant au serveur de la fac.

Malgré les coupures dues aux erreurs que rencontrent la connexion avec le SSH et que rencontrait aussi mes autres camarades de classe, j'ai pu tout de même me rattraper la dernière semaine avant le rendu.



Au niveau du schéma j'ai utilisé un logiciel **DBSCHEMA** fourni par un site open source.

Puis pour mes requêtes, je me suis documenté plus sur le manuel fournis sur Moodle et documentation existante en ligne.

1.8 Les Contraintes :

Toutes les trouver dans le fichier TABLES.SQL

```
/*--------------------------------------------------------------------------
                                CONTRAINTES
--------------------------------------------------------------------------*/
/*user*/
ALTER TABLE utilisateurs
    ADD CONSTRAINT utilisateurs_pk PRIMARY KEY (id_utilisateur);
ALTER TABLE utilisateurs
    ADD CONSTRAINT utilisateurs_u UNIQUE (login);
ALTER TABLE utilisateurs
    ADD CONSTRAINT CK_EMAIL check (email LIKE '%_@_%');

/*emission*/
ALTER TABLE emission
    ADD CONSTRAINT emission_pk PRIMARY KEY (id_emission);
ALTER TABLE emission
    ADD CONSTRAINT emission_nom_emission_u UNIQUE (nom_emission);
ALTER TABLE emission
    ADD CONSTRAINT fk_emission_types FOREIGN KEY (nom_type) REFERENCES types ON DELETE CASCADE;

/*video*/
ALTER TABLE video
    ADD CONSTRAINT video_pk PRIMARY KEY (id_video);
ALTER TABLE video
    ADD CONSTRAINT video_titre_u UNIQUE(titre);
ALTER TABLE video
    ADD CONSTRAINT fk_video_emission FOREIGN KEY (id_emission) REFERENCES emission ON DELETE CA
SCADE;

/*favoris*/
ALTER TABLE favoris
    ADD CONSTRAINT fk_favoris_utilisateurs FOREIGN KEY (id_utilisateur) REFERENCES utilisateurs
;
ALTER TABLE favoris
    ADD CONSTRAINT fk_favoris_video FOREIGN KEY (id_video) REFERENCES video;

/*preferences*/
ALTER TABLE preferences
    ADD CONSTRAINT fk_utilisateurs_types_u FOREIGN KEY (id_utilisateur) REFERENCES utilisateurs
;
ALTER TABLE favoris
    ADD CONSTRAINT fk_utilisateurs_video FOREIGN KEY (nom_type) REFERENCES types;

/*follow*/
ALTER TABLE follow
    ADD CONSTRAINT utilisateurs_follow_pk PRIMARY KEY (id_utilisateur, id_emissin);
```

1.8.1 Tests et requêtes :

Pour mes tables sont toutes remplies, comme vous pouvez le voir avec TESTS et requêtes mises en toute liberté pour assurer de la bonne création des tables // comme vous pouvez le voir, un aperçu des résultats de requêtes

```
wolfdemon@wolfdemon: ~
DESCRIPTION_VIDEO
--------------------------------------------------------------------------------
MUL
---
LOCALISATION
--------------------------------------------------------------------------------
DATE_P_DUFFISIO      DUREE FORMAT_IMAGE
--------------- --------- ------------------------------


15 rows selected.

SQL> select distinct titre from video;

TITRE
--------------------------------------------------------------------------------
dicaprio
le sauvetage de la vache
philosophie de musique
promenade anglaise
france en scicilie
comedie
relativité
brad ou norton
David Finher
la note
la plume de hugo

TITRE
--------------------------------------------------------------------------------
apocalypse
wild et schakespear
revisitr theatre
music gladiator

15 rows selected.

SQL>
```

```
wolfdemon@wolfdemon: ~
SQL> select id_utilisateur, count(*) AS nbhistorique from historique group by id_utilisateur;

ID_UTILISATEUR NBHISTORIQUE
-------------- ------------
             1            4
             6            1
             2            3
             4            1
             5            1
             8            1
             3            1
             7            2
            10            1
             9            1

10 rows selected.
```

Pour les requêtes demandées : j'ai répondu à trois dont y a un bon retour, les deux autres je ne suis pas trop si sûr.

```
SQL> select nom_type,count(*) from historique,video,emission,types where historique.id_video=video.id_video and video.id_emission=emission.id_
emission and emission.id_type=types.id_type group by nom_type;

NOM_TYPE
--------------------------------------------------------------------------------
  COUNT(*)
----------
music
         1

litterature
         1

nature
         1


NOM_TYPE
--------------------------------------------------------------------------------
  COUNT(*)
----------
science
         1

cinema
         6

SQL>
```

Pour celle de différence absolue de visionnage entre allemand et français…

```
SQL> select nationalite,video.id_video,count(*)as compteur from utilisateurs ,historique , video where utilisateurs.id_utilisateur=historique.
id_utilisateur and video.id_video= historique.id_video and (nationalite='france' or nationalite='allemagne') group by nationalite,video.id_vid
eo order by abs (compteur);

NATIONALITE
--------------------------------------------------------------------------------
  ID_VIDEO   COMPTEUR
---------- ----------
france
         3          1

allemagne
         5          1

allemagne
        15          1


NATIONALITE
--------------------------------------------------------------------------------
  ID_VIDEO   COMPTEUR
---------- ----------
allemagne
         7          2

france
         1          3

france
         7          4

6 rows selected.

SQL>
```

2 PARTIE 2 :

2.1 Les modifications de tables :

Pour cette deuxième partie il a fallu apporter quelques modifications, en rajoutant la nouvelle table VideoArchivee :

```
CREATE TABLE  videoArchivee
(
    id_video        NUMBER NOT NULL,
    id_emission     integer not null,
    titre           VARCHAR2(100),
    description_video     VARCHAR2(500),
    multilangue         CHAR(1) DEFAULT '0',
    localisation    VARCHAR2(250),
    date_p_duffision      DATE NOT NULL,
    duree    NUMBER NOT NULL,
    format_image    VARCHAR(10) NOT NULL
);
```

Et rajouter l'attribut nombredevideoaimee dans la table utilisateurs, qui a une contrainte avec chiffre appartenant à l'intervalle [0, 300]

```
CREATE TABLE utilisateurs
(
    id_utilisateur          NUMBER NOT NULL,
    login                   VARCHAR2(60) NOT NULL,
    motdepass               VARCHAR2(60) NOT NULL,
    nom                     VARCHAR2(20),
    prenom                  VARCHAR2(20),
    email                   VARCHAR2(50) NOT NULL,
    dateNaissance           DATE,
    newsletter              CHAR(1) DEFAULT '0',
    nombreDeVideoAimee      NUMBER NOT NULL,
    nationalite             VARCHAR(30)
);
```

```
ALTER TABLE utilisateurs
    ADD CONSTRAINT CK_nombreDeVideoAimee CHECK((nombreDeVideoAimee
>= 0) AND (nombreDeVideoAimee < 301));
```

2.2 Travail demandé :

2.3 Procédure ph/sql

2.3.1 Conversion en JSON :

J'ai dû le faire avec un JSON_Object qui relie les informations vidéo au format JSON ainsi j'ai rajouté une autre méthode plus simpliste qui n'a rien à avoir avec une fonction et que je trouve plus pratique, et que j'ai voulu mentionner.

2.3.2 La génération d'un texte pour la newsletter :

J'ai utilisé un curseur sur la table vidéo, pour créer une liste, dans laquelle je retourne les vidéos dont la date de diffusion n'est pas loin d'une semaine de la date actuelle.

2.3.3 Les vidéos les plus populaires :

2.4 Contraintes d'intégrité du projet :

2.4.1 Limiter le nombre de favoris à 300

```
CREATE OR REPLACE TRIGGER checkLimiteFavoris
  BEFORE INSERT ON favoris
  FOR EACH ROW
    declare
    nb_videoliked INTEGER;
  BEGIN
    SELECT count(id_utilisteur) into nb_videoliked FROM favoris
    WHERE id_utilisateur = :new.id_utilisateur
    group by id_utilisateur

  IF nb_videolikee >= 300 THEN
        RAISE_APPLICATION_ERROR(-20004, 'IMPOSSIBLE D AJOUTER UNE
VIDEO AU FAVORIS,IL Y A 300 DEJA, ESPACE REMPLI!!!');
  END IF;
END;
/
SHOW ERRORS
```

2.4.2 La mise a jour de de date :

A voir comment c'est réalisé sur le fichier, avec la condition des du temps, fine diffusion consiste à la durée.

2.4.3 La suppression vidéo et l'archivage :

C'est fait sur le fichier à voir.

2.4.4 Gestion de spam :

J'ai défini deux variable (tempsactuel)timestp et nombre_visionnage

```
CREATE OR REPLACE TRIGGER visionage_controle
  BEFORE INSERT ON historique
  FOR EACH ROW
  DECLARE
    delai INTEGER;
    date_vis DATE;
  BEGIN
    SELECT max(date_visionnage) INTO date_vis FROM historique WHERE
id_utilisateur = :new.id_utilisateur;
    delai := SYSDATE - date_vis
    IF delai >=0  THEN
      IF (timestp - current_timestamp < interval '1' minute)THEN
        IF nombre_visionage< 3 THEN
            INSERT INTO historique VALUES (id_historique, id_video,
id_utilisateur,current_date);
        ELSE
        RAISE_APPLICATION_ERROR(-20004, 'Vous nepouvez pas regarder
3 videos en une minutes, attendez!');
         END IF;
      ELSE
        nombre_visionage = 1;
        INSERT INTO historique VALUES (id_historique, id_video,
id_utilisateur,current_date);
      END IF;
    END IF;
END;
/
SHOW ERRORS
```

2.5 Sources :

J'ai pu m'inspirer de ces modèles pour mieux comprendre les curseurs et pouvoir l'appliquer, ainsi des tp réalisés durant nos séances organisées.

http://salihayacoub.com/420Keh/Semaine%203/PLSQLIntro.pdf

https://www.emi.ma/ntounsi/COURS/DB/Polys/SQL/Exercices/PL-SQL-Exemples.html#mozTocId259058

https://stackoverrun.com/fr/q/3696063

https://www.fichier-pdf.fr/2014/06/02/pl-sql-exercices-corriges/

http://courssql.com/cours-sql-oracle-07-04-cursor-curseur