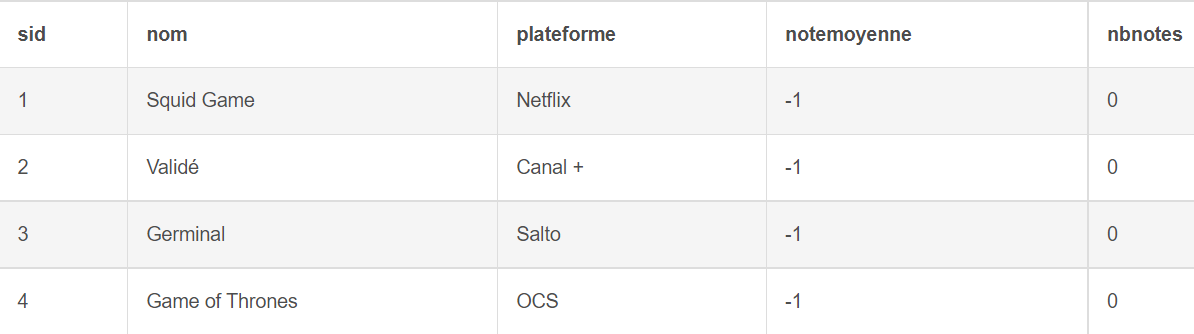
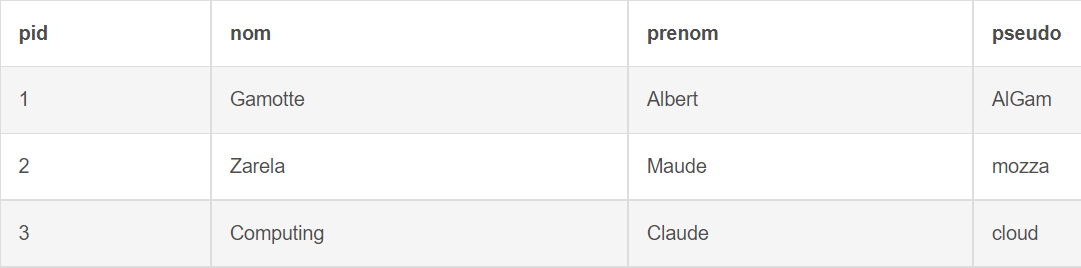
Projet de Bases de Données Relationnelles

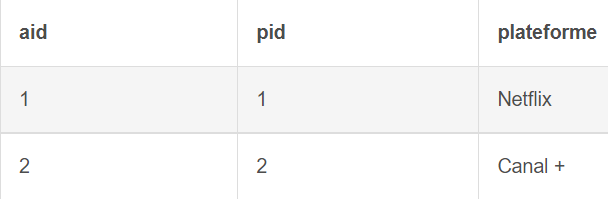
# Enoncé

Le script1 SQL, situé à la fin du sujet, modélise une base de données permettant de gérer les évaluations de séries télévisées.

Une série télévisée est décrite par un nom et est diffusée par une plateforme. Deux séries ne peuvent pas avoir le même nom. Un exemple d’instance (créée à partir du script situé en fin de sujet) de la relation serie est :

Une personne est caractérisée par un nom, un prénom et un pseudo. Il n’est pas possible d’avoir 2 pseudos identiques. Un exemple d’instance (créée à partir du script situé en fin de sujet) de la relation personne est :



Une personne peut être abonnée ou non à une plateforme. Un exemple d’instance (créée à partir du script situé en fin de sujet) de la relation abonnes est :

Une personne peut donner une note (entre 0 et 5) à chaque série, qu’elle soit abonnée ou non à la plateforme qui diffuse la série. Pour chaque série, la base enregistre la note moyenne des notes et le nombre de notes. Un exemple d’instance (créée à partir du script situé en fin de sujet) de la relation evaluation est :

# Travail à réaliser

.

## Finalisation du script SQL

Vous devez **ajouter toutes les contraintes nécessaires à la bonne gestion de la base**

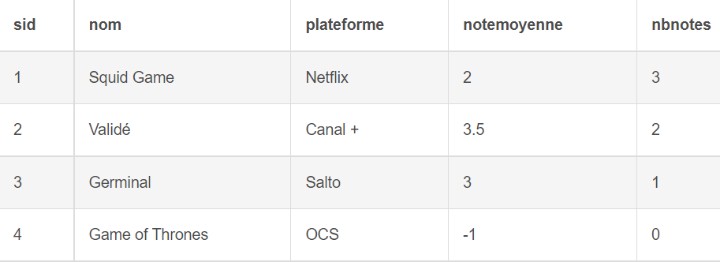
de données (clé primaire, clé étrangère, contraintes de domaine, d’unicité etc.).

Vos contraintes doivent permettre d’exécuter les requêtes d’insertion données à la fin du script. Il est à noter que **la suppression d’un nuplet dans personne doit entraîner la suppression de ses évaluations et de ses abonnements**.

## Développement d’un déclencheur

Vous implémenterez un **déclencheur** (*trigger*) en PL/pgSQL sous PostgreSQL, qui, lorsqu’on insère une note dans la relation evaluation*,* met à jour la note moyenne et le nombre de notes de la série associée.

Par exemple, après l’exécution des insertions dans la relation évaluation (cf. script situé à la fin du sujet), la relation serie doit contenir les nuplets suivants :



## Requêtes d'interrogation en SQL

*Vous pouvez insérer, dans la base, des nuplets supplémentaires, pour tester les requêtes.*

## Ecrivez en SQL les requêtes suivantes3 :

* 1. Quelles plateformes n’ont pas d’abonnés ?

Les PF sans abonnés = toutes PF des séries – les PF des abonnés

ΠPF(σPF-> s1.PF(Serie)) - ΠPF(σPF-> a1.PF(Abonnes))

* 1. Quelles personnes (en donnant son pseudo) a évalué une série de Netflix ou une série de Canal+ ? ***Écrire 2 requêtes SQL, une avec UNION et une sans UNION et comparer leur temps d'exécution*.**

Πpseudo (σPF=’Netflix’ ∨ ‘Canal +’ ((σsid->s.sid(Serie)) ⊗s.sid=e.sid (Πpseudo,e.sid ((σpid->p.pid(Personne)) ⊗a.pid=e.pid (σpid>e.pid(Evaluation)))

Autre forme

A =Πpseudo (σPF=’Netflix’ ((σsid->s.sid(Serie)) ⊗s.sid=e.sid (Πpseudo,e.sid ((σpid->p.pid(Personne)) ⊗a.pid=e.pid (σpid>e.pid(Evaluation)))

B=Πpseudo (σPF=’Canal +’ ((σsid->s.sid(Serie)) ⊗s.sid=e.sid (Πpseudo,e.sid ((σpid->p.pid(Personne)) ⊗a.pid=e.pid (σpid>e.pid(Evaluation)))

A∪B

La requête avec le OR exécutée en 45 ms contre 50 ms pour l’union

* 1. Quelles personnes (en donnant son pseudo) a évalué une série de Netflix **et** une série de Canal+ ? ***Écrire 2 requêtes SQL, une avec INTERSECT et une sans INTERSECT et comparer leur temps d'exécution***.

Rappel intersection de A & B = A**-(A-B) noter A∩B**

A =idem question b

B =idem question b

**A∩B** (intersection correspond à un et noté ∧)

Requête sans intersect en 49 ms et avec en 44 ms

* 1. Quelles séries (en donnant leur nom) ont été évaluées par au moins 2 personnes ? ***Écrire 3 requêtes SQL, une requête sans GROUP BY et sans EXISTS, une requête sans GROUP BY mais avec un EXITS et une requête avec GROUP BY et sans EXITS. Comparer leur temps d'exécution***.

C=((σsid->e2.sid(Evaluation) ⊗e2.sid=e.1sid (σsid->e1.sid(Evaluation))

D=(Πe1.sid (σe2.pid !=e1.pdd (C)))

E= Πpseudo (σsid->s.sid(Série)) ⊗s.sid=e1.sidD

Temps execution requete sasn group by et sans exists = 38 ms

Temps execution requete sasn group by et avec exists = 46 ms

Temps execution requete avec group by et sans exists = 42 ms

-----1iere forme sans group by

* 1. Quelles séries (en donnant leur nom) ont été évaluées par toutes les personnes de la base de données ? ***Écrire 2 requêtes SQL, une requête sans GROUP BY et une requête avec GROUP BY et comparer leur temps d'exécution***.

Séries telles qu’il n’existe pas de personnes qui ne l’aient pas toutes évalué

Etape 1 séries et les personnes qui ne les ont pas évaluées

A=Πpid,sid(Evaluation) -> les séries évaluées (avec les personnes qui les ont évaluées)

B=Πpid(Personnes) x Πsid (Serie)-> ttes les combinaisons de series et de personnes

C=B-A -> toutes les combinaisons de series et des personnes qui ne les ont pas évaluées

C’=Πsid(Πpid(Personnes) x Πsid (Serie)- Πpid,sid(Evaluation))

Πnom (Serie)-C’ -> ttes les séries moins

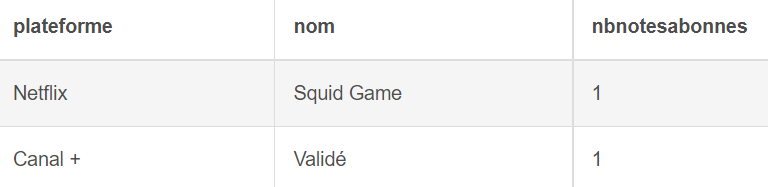
* 1. Quelles séries (en donnant leur nom) sont les moins bien notées ? (*Vous ne devez tenir compte que des séries ayant au moins une note dans la relation* evaluation)

Non demandé

A=Πs.sid,nom(Serie) Θ s.sid=e1.sid Πe1.sid ((σsid -> e1.sid (Evaluation)) -> ttes les series évaluées

B=Πe2.sid,nom(Serie) Θ s.sid=e2.sid Πe2.sid ((σe2.sid>e1.sid (σsid -> e2.sid (Evaluation)) -> ttes les series évaluées avec une note strictement supérieure aux notes de l’évaluation E1

A-B -> reste ttes les notes inférieures ou égales

* 1. Quel est le nombre de notes par série en ne tenant compte que des notes données par les abonnés de la plateforme diffusant la série ? *Par exemple, sur les nuplets insérés par le script donné en fin de sujet, votre requête doit renvoyer :*

3 Vous pouvez utiliser des vues pour décomposer vos requêtes en sous-requêtes si besoin.

* 1. Quel est le nombre de notes par série en séparant les notes données par les abonnés de la plateforme diffusant la série et les notes des personnes non abonnées à la plateforme ? *Par exemple, sur les nuplets insérés par le script donné en fin de sujet, votre requête doit renvoyer :*

## Requêtes d'interrogation en Algèbre relationnelle

Ecrire les requêtes a) à e) en Algèbre Relationnelle. *Pour simplifier vous pouvez écrire les opérateurs algébriques en français (sélection, projection, jointure, division etc.).*

# Rendu du TP

Ce TP donnera lieu à un **script SQL** contenant :

* La création du schéma de la base de données corrigé (avec l’ajout des contraintes).
* Le code du déclencheur. *A défaut (si votre déclencheur ne fonctionne pas), vous devez indiquer en français ce que doit faire le déclencheur et quelles requêtes SQL il doit contenir.*
* Le script d'insertion des nuplets permettant de tester les requêtes et le déclencheur.
* Les requêtes d’interrogation en SQL. *Merci d’indiquer svp en commentaire l’intitulé de chaque requête.*
* Les requêtes a) à e) en Algèbre Relationnelle.

Le **script SQL devra être enregistré dans un fichier,** dont le nom devra obligatoirement être de la forme **Nom1Prenom1Nom2Prenom2.sql** (i.e. en précisant les noms et prénoms des 2 membres du binôme), et **devra être déposé sur Moodle.**

## Script SQL (incomplet) de création des relations de la base de données :

CREATE TABLE serie (

SID serial,

nom varchar(25), plateforme varchar(25),

noteMoyenne float DEFAULT -1, nbNotes integer DEFAULT 0

);

CREATE TABLE personne ( PID serial,

nom varchar(25), prenom varchar(25), pseudo varchar(25)

);

CREATE TABLE evaluation ( EID serial,

PID integer, SID integer,

note integer DEFAULT 5

);

CREATE TABLE abonnes ( AID serial,

PID integer, plateforme varchar(25)

);

## Script SQL d’insertion des nuplets :

INSERT INTO Serie(nom,plateforme) VALUES ('Squid Game','Netflix'), ('Validé','Canal

+'),('Germinal','Salto'),('Game of Thrones','OCS');

INSERT INTO Personne(nom,prenom,pseudo) VALUES ('Gamotte','Albert','AlGam'),('Zarela','Maude','mozza'),('Comp uting','Claude','cloud');

INSERT INTO evaluation (PID,SID,note) VALUES (1,1,4),(1,2,4),(1,3,3),(2,1,0),(2,2,3),(3,1,2);

INSERT INTO abonnes(PID,plateforme) VALUES (1,'Netflix'), (2,'Canal +');