

**SAE204 S2 2023**

**Rapport final**

# Sommaire

Notre choix de série

Les questions

Modèle conceptuel de donnée

Schéma relationnel

Les triggers

Les vues

Les fonctions

Les procédures

Les créations de tables

Les insertions de données

Répartition du travail

# Notre choix

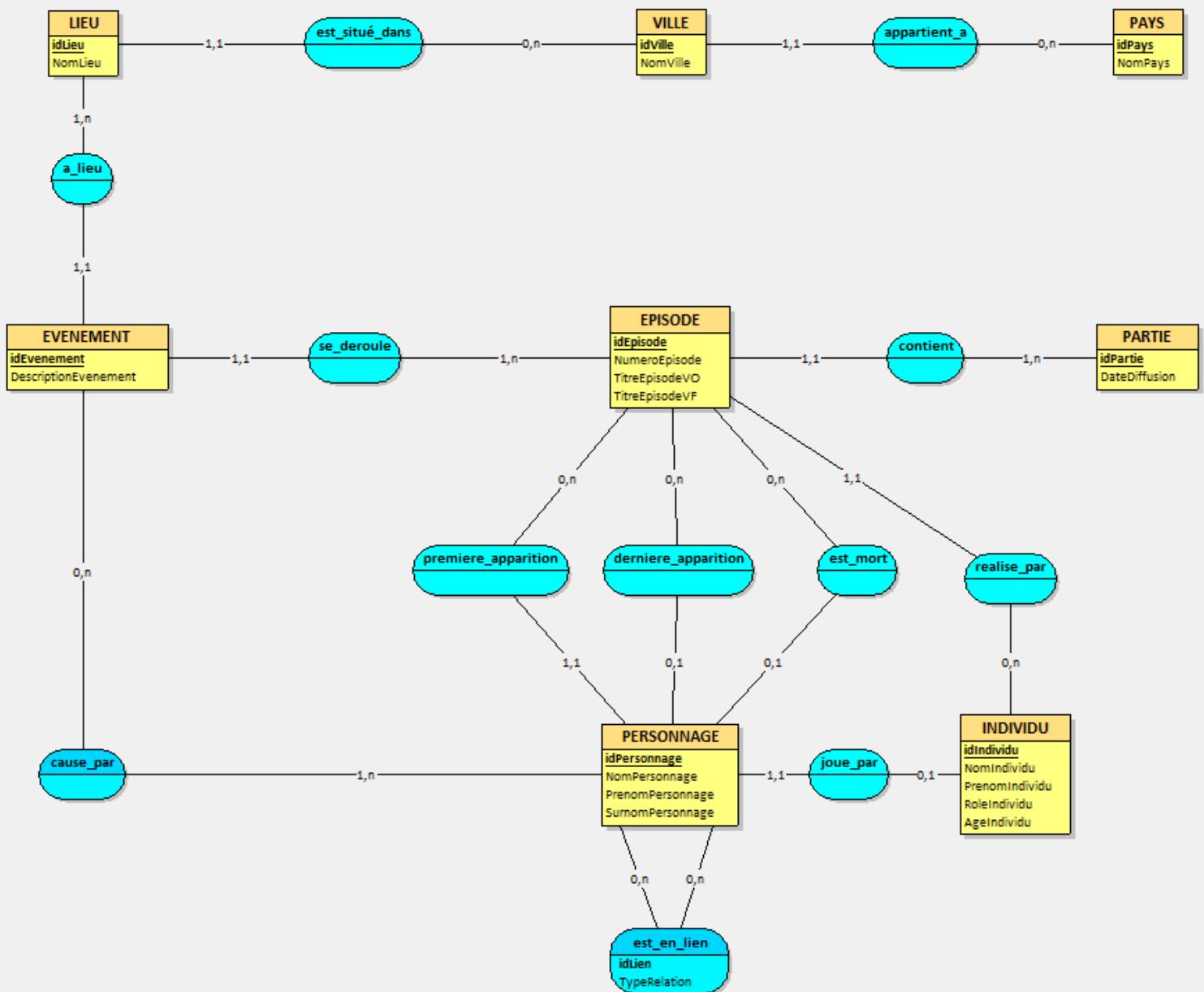
Nous avons choisi une série intitulée La Casa De Papel. Cette série est une série espagnole créée par Alex Pina et diffusée depuis le 02 mai 2017 sur la chaîne Atena 3 en Espagne. Tous les membres du groupe connaissaient cette série grâce à la plateforme de streaming Netflix. En effet, cette série connut pour ses musiques emblématiques rassemble des millions de foyers et des milliers de téléspectateurs. La Casa De Papel, composé de cinq parties, raconte l'histoire d'un homme mystérieux surnommé "Professeur" prépare le plus grand braquage de tous les temps. Pour mener à bien son plan, il recrute huit des meilleurs criminels espagnols, qui n'ont rien à perdre. Leur objectif était d'infiltrer l'usine de monnaie nationale, d'imprimer 2,4 milliards d'euros en petites coupures de 50 euros, et ce, en moins de 11 jours sans faire de victimes, malgré la présence de 67 otages, dont la fille de l'ambassadeur de France. Grande-Bretagne, Alison Parker, il n'y avait pas de relation amoureuse entre les braqueurs... Le professeur a également demandé aux membres du gang de choisir un pseudonyme parmi les noms des villes : Tokyo, Rio, Berlin, Denver, Nairobi, Oslo, Helsinki et Moscou. Dans la deuxième saison (composée des parties 3, 4 et 5), le professeur fait revivre un objet que son frère avait imaginé avant le premier casse. Leur objectif est d'infiltrer la Banque d'Espagne, de voler 90 tonnes d'or et de sauver son complice Rio, qui est emprisonné et torturé illégalement par la police espagnole, qui garde ses opérations secrètes.

Cette série est pleine de rebondissement et de données. C'est la raison pour laquelle La Casa De Papel est un très bon sujet de base de données pouvant aider les réalisateurs de celle-ci.

# Nos questions

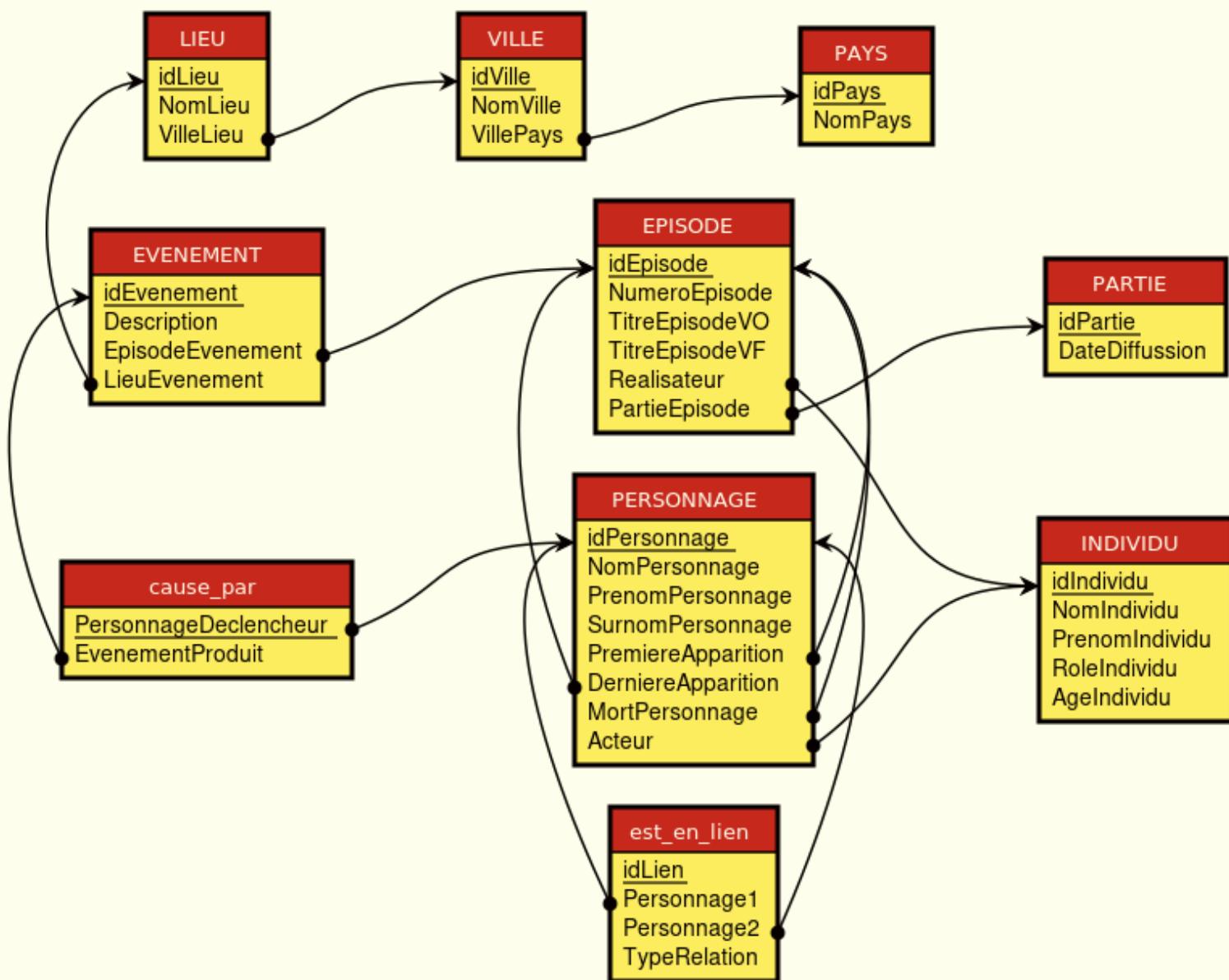
1. Quel est l'âge de l'actrice de Lisbonne ?
2. Quels sont les lieux que Marquina Sergio a visité ?
3. Combien d'évènement se déroule dans la partie 1 ?
4. Combien d'évènement à causé le personnage de Raquel Murillo durant l'épisode de son apparition ?
5. Quels sont les évènements causé par le personnage de Raquel Murillo ?
6. Quels sont les personnages qui ne sont pas réapparu ?
7. Quels sont les acteurs qui sont présent dans la partie 2 ?
8. Combien d'évènement a causé le personnage Murillo à la Banque d'Espagne ?
9. Quelle est la liste des réalisateurs de la partie 3 ?
10. Quelle est la relation entre les personnages Tokyo et Rio ?
11. Dans quel partie apparaît le personnage Tokyo ?
12. Combien de personnage sont mort dans la partie 2 ?
13. Quels sont les évènements causé dans le lieu à Madrid ?
14. Quels sont les personnages reliés au personnage Rio ?
15. Dans quel épisode le personnage Tokyo est mort ?
16. Quels sont les évènements causé dans la ville Madrid ?
17. Combien y a-t-il d'épisode dans la partie 2 ?
18. Quels sont les épisodes que le réalisateur Quintas Javier a réalisé ?
19. Quels sont les personnages en relation de type Coéquipiers?
20. Combien d'épisode a réalisé Rodiguo Alex ?

# Model Conceptuel de données



Nous avons conçu notre Modèle Conceptuel de Données tous ensemble. Lors de notre réflexion pour la conception de ce Modèle Conceptuel de Données, nos tables centrales étaient PERSONNAGE, EPISODE et EVENEMENT. Nous avons développé les autres tables par la suite. Tout au long de la conception, nous avons porté une attention particulière au respect des normes : titres des tables en majuscules, aucun pluriel, aucun attribut identique, attribut avec des majuscules et minuscules, titres de tables d'associations en minuscule avec des verbes, et les clés primaires sont précédés d'id. Nous avons veillé à éviter les données répétitives afin d'optimiser le stockage. C'est la raison pour laquelle les lieux des événements sont décomposés en trois tables. Étant donné que plusieurs lieux peuvent se dérouler dans une même ville et des villes peuvent être dans les mêmes pays. Nous avons décomposé en trois tables : LIEU, VILLE et PAYS. Vous pourrez remarquer une subtilité dans nos associations de tables entre PERSONNAGE et EPISODE. En effet, un personnage possède une première apparition dans la série. Mais aussi une dernière apparition et une mort. La différence entre la dernière apparition et la mort du personnage est subtile. La Casa De Papel contient des flash-back, alors de personnage comme Berlin sont mort au début de la série, mais réapparaît tout au long de la série, même après sa mort.

# Schéma Relationnel



Notre schéma relationnel reprend notre modèle conceptuel de données. Nous avons continué à respecter les mêmes normes et avons modifié le nom des clés étrangères par des noms plus appropriés. De plus, nous avons choisi de faire le schéma relationnel sur Mocodo Online afin d'avoir un schéma relationnel plus précis qu'un schéma relationnel généré par le logiciel Looping. Voici le script saisi pour la création du schéma relationnel sur Mocodo Online :

```
LIEU : idLieu, NomLieu, VilleLieu->VILLE->idVille
:
VILLE : idville, NomVille, villePays->PAYS->idPays
:
PAYS : idPays, NomPays
:
:
EVENEMENT : idEvenement, Description, EpisodeEvenement->EPISODE->idEpisode, LieuEvenement->LIEU->idLieu
:
:
EPISODE : idEpisode, NumeroEpisode, TitreEpisodeVO, TitreEpisodeVF, Realisateur->INDIVIDU->idIndividu,
PartieEpisode->PARTIE->idPartie
:
:
PARTIE : idPartie, DateDiffussion
:
:
cause_par : PersonnageDeclencheur->PERSONNAGE->idPersonnage, EvenementProduit->EVENEMENT->idEvenement
:
:
PERSONNAGE : idPersonnage, NomPersonnage, PrenomPersonnage, SurnomPersonnage, PremiereApparition->EPISODE-
>idEpisode, DerniereApparition->EPISODE->idEpisode, MortPersonnage->EPISODE->idEpisode, Acteur->INDIVIDU->idIndividu
:
:
INDIVIDU : idIndividu, NomIndividu, PrenomIndividu, RoleIndividu, AgeIndividu
:
:
est_en_lien : idLien, Personnage1->PERSONNAGE->idPersonnage, Personnage2->PERSONNAGE->idPersonnage, TypeRelation
```

# Nos triggers

```
--Déclencheur permettant de vérifier l'identité des auteurs de changement sur les tables --
CREATE OR REPLACE TRIGGER check_user_permission_partie
BEFORE INSERT OR UPDATE ON PARTIE
FOR EACH ROW
BEGIN
    IF USER NOT IN ('eboissl', 'rbourde', 'cmarti23', 'mpereil6') THEN
        RAISE_APPLICATION_ERROR(-20011, 'Permissions non accordée');
    END IF;
END;
/
CREATE OR REPLACE TRIGGER check_user_permission_pays
BEFORE INSERT OR UPDATE ON PAYS
FOR EACH ROW
BEGIN
    IF USER NOT IN ('eboissl', 'rbourde', 'cmarti23', 'mpereil6') THEN
        RAISE_APPLICATION_ERROR(-20012, 'Permissions non accordée');
    END IF;
END;
/
CREATE OR REPLACE TRIGGER check_user_permission_personnage
BEFORE INSERT OR UPDATE ON PERSONNAGE
FOR EACH ROW
BEGIN
    IF USER NOT IN ('eboissl', 'rbourde', 'cmarti23', 'mpereil6') THEN
        RAISE_APPLICATION_ERROR(-20013, 'Permissions non accordée');
    END IF;
END;
/
```

Ce déclencheur permet vérifier l'identité des individus qui insèrent ou modifient les tables de notre base de données.

Celui-ci a pour objectif d'empêcher les insertions d'épisodes dans la partie 1 à 4 car elles sont terminées.

```
-- Déclencheur permettant d'interdire les insertions d'épisodes dans les parties 1 à 4 --
CREATE OR REPLACE TRIGGER check_partie_limit
BEFORE INSERT ON EPISODE
FOR EACH ROW
DECLARE
    partie_id NUMBER(5);
BEGIN
    SELECT PartieEpisode INTO partie_id FROM PARTIE WHERE idPartie = :new.PartieEpisode;
    IF partie_id BETWEEN 1 AND 4 THEN
        RAISE_APPLICATION_ERROR(-20020, 'L''insertion d''épisodes dans les parties de 1 à 4 est interdite.');
    END IF;
END;
/
```

```
--Trigger permettant de lancer une alerte si une modification incohérente de l'âge d'un individu est essayée--
CREATE OR REPLACE TRIGGER Alerte_Age
BEFORE UPDATE OF AgeIndividu
ON INDIVIDU FOR EACH ROW
BEGIN
    IF :new.AgeIndividu < (:old.AgeIndividu)
        THEN dbms_output.put_line('L''âge est incohérent.');
    END IF;
END;
/
```

Celui-ci vérifie que la modifications de l'âge d'un individu est cohérente. C'est-à-dire que le nouveau âge inséré ne soit pas inférieur au précédent.

```
--Trigger permettant d'empêcher la suppression d'un individu dans la table INDIVIDU s'il est associé à des épisodes en tant que réalisateur--
CREATE OR REPLACE TRIGGER verif_suppression_realisateur
BEFORE DELETE ON INDIVIDU
FOR EACH ROW
DECLARE
    existing_episodes NUMBER(5);
BEGIN
    SELECT COUNT(*) INTO existing_episodes
    FROM EPISODE
    WHERE RealisateurEpisode = :old.idIndividu;

    IF existing_episodes > 0 THEN
        RAISE_APPLICATION_ERROR(-20021, 'Impossible de supprimer l''individu car il est associé en tant que réalisateur à des épisodes.');
    END IF;
END;
/
```

Ce trigger empêchant d'empêcher l'insertion d'un nouvel épisode si le réalisateur n'est pas un individu existant dans la table INDIVIDU.

Ce trigger permet d'interdire la suppression d'un individu s'il est relié à un épisode en tant que réalisateur.

```
--Trigger permettant d'empêcher l'insertion d'un nouvel épisode si le réalisateur n'est pas un individu existant dans la table INDIVIDU--
CREATE OR REPLACE TRIGGER verif_realisateur_existant
BEFORE INSERT ON EPISODE
FOR EACH ROW
DECLARE
    existing_director NUMBER(5);
BEGIN
    SELECT COUNT(*) INTO existing_director
    FROM INDIVIDU
    WHERE idIndividu = :new.RealisateurEpisode;

    IF existing_director = 0 THEN
        RAISE_APPLICATION_ERROR(-20022, 'Le réalisateur spécifié n''existe pas dans la table INDIVIDU.');
    END IF;
END;
/
```

```
--Trigger permettant d'empêcher l'insertion d'un nouveau personnage si l'acteur n'est pas un individu existant dans la table INDIVIDU--
CREATE OR REPLACE TRIGGER verif_personnage_existant
BEFORE INSERT ON PERSONNAGE
FOR EACH ROW
DECLARE
    existing_character NUMBER(5);
BEGIN
    SELECT COUNT(*) INTO existing_character
    FROM INDIVIDU
    WHERE idIndividu = :new.Acteur;

    IF existing_character = 0 THEN
        RAISE_APPLICATION_ERROR(-20023, 'L''acteur spécifié n''existe pas dans la table INDIVIDU.');
    END IF;
END;
/
```

Ce déclencheur permet d'empêcher la suppression d'un pays tant qu'il est associé à des villes.

Le déclencheur permettant d'empêcher l'insertion d'un nouveau personnage si l'acteur n'est pas un individu existant dans la table INDIVIDU

```
--Ce déclencheur permet d'empêcher la suppression d'un pays tant qu'il est associé à des villes--
CREATE OR REPLACE TRIGGER verif_pays_utilise
BEFORE DELETE ON PAYS
FOR EACH ROW
DECLARE
    existing_usage NUMBER(5);
BEGIN
    SELECT COUNT(*) INTO existing_usage
    FROM VILLE
    WHERE idPays = :old.idPays;

    IF existing_usage > 0 THEN
        RAISE_APPLICATION_ERROR(-20024, 'Impossible de supprimer le pays car il est utilisé par une ou plusieurs villes.');
    END IF;
END;
/
```

# Nos vues

Cette vue permet de voir toutes les relations entre les personnages :

```
CREATE OR REPLACE view Relations_personnage (idPersonnage1, SurnomPerso1 ,PrenomPerso1, NomPerso1, idPersonnage2, SurnomPerso2, PrenomPerso2, NomPerso2, TypeRelation)AS(
    SELECT DISTINCT Personnage1, P1.SurnomPersonnage, P1.PrenomPersonnage, P1.NomPersonnage, Personnage1, P2.SurnomPersonnage, P2.PrenomPersonnage, P2.NomPersonnage, TypeRelation
    FROM est_en_lien
    INNER JOIN Personnage P1 ON P1.idPersonnage = Personnage1
    INNER JOIN Personnage P2 ON P2.idPersonnage = Personnage2
);
```

Cette vue a pour objectif de voir quels acteurs interprètent quel personnage :

```
CREATE OR REPLACE view Acteur_personnage (NomActeur, PrenomActeur, NomPersonnage, PrenomPersonnage, SurnomPersonnage)AS(
    SELECT Nomindividu,PrenomIndividu,SurnomPersonnage,PrenomPersonnage,NomPersonnage
    FROM INDIVIDU INNER JOIN PERSONNAGE ON INDIVIDU.idIndividu = PERSONNAGE.idPersonnage
);
```

Cette vue a pour objectif de voir quel réalisateurs réalise quel épisode :

```
CREATE OR REPLACE VIEW Realisateur_De (NomRealisateur, PrenomRealisateur, EpisodeRealise, Partie)
AS (
    SELECT Nomindividu, PrenomIndividu, NumeroEpisode, PartieEpisode
    FROM INDIVIDU
    INNER JOIN EPISODE ON INDIVIDU.idIndividu = EPISODE.idEpisode
    INNER JOIN PARTIE ON EPISODE.PartieEpisode = PARTIE.idPartie
);
```

Grâce à cette vue, nous pouvons apercevoir quels sont les personnages non réapparus et quand ils ont disparu :

```
CREATE OR REPLACE VIEW PersonnageDisparu (NomPersonnage, PrenomPersonnage, EpisodeDisparu, Partie)
AS (
    SELECT PERSONNAGE.NomPersonnage, PERSONNAGE.PrenomPersonnage, PERSONNAGE.DerniereApparition, EPISODE.PartieEpisode
    FROM PERSONNAGE
    INNER JOIN EPISODE ON PERSONNAGE.DerniereApparition = EPISODE.idEpisode
    WHERE PERSONNAGE.DerniereApparition < 30
    OR PERSONNAGE.DerniereApparition IS NULL
);
```

# Nos fonctions

Cette fonction retourne le nombre qu'a réalisé un individu. Elle retourne une exception s'il n'y aucun épisode.

```
CREATE OR REPLACE FUNCTION nbEpisode(NomIndividuRecherche INDIVIDU.Nomindividu%TYPE, PrenomIndividuRecherche INDIVIDU.Prenomindividu%TYPE)
RETURNS NUMBER IS
    nbEpisode NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(DISTINCT EPISODE.idEpisode) INTO nbEpisode
    FROM EPISODE
    INNER JOIN INDIVIDU ON EPISODE.Realisateur = INDIVIDU.idIndividu
    WHERE INDIVIDU.nomindividu LIKE (NomIndividuRecherche)
    AND INDIVIDU.prenomindividu LIKE (PrenomIndividuRecherche);
    IF nbEpisode<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbEpisode;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucun épisode.');
    RETURN nbEpisode;
END;
/
```

Cette fonction retourne le nombre de relation d'un personnage selon son surnom. Si le personnage n'a aucune relation, une exception est relevée.

```
CREATE OR REPLACE FUNCTION nbRelation(SurnomPersonnageRecherche PERSONNAGE.SurnomPersonnage%TYPE)
RETURNS NUMBER IS
    nbRelation NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(est_en_lien.idLien) INTO nbRelation
    FROM est_en_lien
    INNER JOIN PERSONNAGE p1 ON p1.idPersonnage = est_en_lien.Personnage1
    INNER JOIN PERSONNAGE p2 ON p2.idPersonnage = est_en_lien.Personnage2
    WHERE p1.SurnomPersonnage LIKE (SurnomPersonnageRecherche);
    IF nbRelation<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbRelation;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucune relation');
    RETURN nbRelation;
END;
/
```

Cette fonction retourne le nombre d'évènement se déroulant dans un lieu. Elle retourne une exception s'il n'y aucun évènement.

```
CREATE OR REPLACE FUNCTION nbEvenement(NomLieuCherche LIEU.NomLieu%TYPE)
RETURN NUMBER is
    nbEvenement NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(EVENEMENT.idEvenement) INTO nbEvenement
    FROM EVENEMENT
    INNER JOIN LIEU ON LIEU.idLieu = EVENEMENT.LieuEvenement
    WHERE Lieu.NomLieu LIKE(NomLieuCherche);
    IF nbEvenement<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbEvenement;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucun évènement');
    RETURN nbEvenement;
END;
/
```

Cette fonction retourne le nombre d'évènement se déroulant dans une partie. Elle retourne une exception s'il n'y aucun évènement.

```
CREATE OR REPLACE FUNCTION nbEvenement2(PartieCherche EPISODE.PartieEpisode%TYPE)
RETURN NUMBER is
    nbEvenement NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(EVENEMENT.idEvenement) INTO nbEvenement
    FROM EVENEMENT
    INNER JOIN EPISODE ON EVENEMENT.EpisodeEvenement = EPISODE.idEpisode
    WHERE EPISODE.PartieEpisode LIKE(PartieCherche);
    IF nbEvenement<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbEvenement;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucun évènement');
    RETURN nbEvenement;
END;
/
```

Cette fonction retourne le nombre d'évènement causé par un personnage selon son nom. Elle retourne une exception s'il n'y aucun évènement.

```
CREATE OR REPLACE FUNCTION nbEvenement3(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE)
RETURN NUMBER is
    nbEvenement NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(EVENEMENT.idEvenement) INTO nbEvenement
    FROM EVENEMENT
    INNER JOIN EPISODE ON EPISODE.idEpisode = EVENEMENT.EpisodeEvenement
    INNER JOIN PERSONNAGE ON EPISODE.idEpisode = PERSONNAGE.PremiereApparition
    WHERE PERSONNAGE.NomPersonnage LIKE (NomPersonnageCherche);
    IF nbEvenement<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbEvenement;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucun évènement');
    RETURN nbEvenement;
END;
/
```

Cette fonction retourne le nombre de mort. Elle retourne une exception s'il n'y aucun évènement.

```
CREATE OR REPLACE FUNCTION NbMort(Partie PARTIE.idPartie%TYPE)
RETURN NUMBER is
    nbMort NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(PERSONNAGE.idPersonnage) INTO nbMort
    FROM PERSONNAGE
    INNER JOIN EPISODE ON EPISODE.idEpisode = PERSONNAGE.MortPersonnage
    INNER JOIN PARTIE ON EPISODE.PartieEpisode = PARTIE.idPartie
    WHERE PARTIE.idPartie = Partie;
    IF nbMort<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbMort;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucun mort');
    RETURN nbMort;
END;
/
```

Cette fonction retourne le nombre d'épisode dans une partie. Elle retourne une exception s'il n'y aucun épisode.

```
CREATE OR REPLACE FUNCTION nbEpisode_Partie(Partie PARTIE.idPartie%TYPE)
RETURN NUMBER is
    nbEpisode NUMBER(5) := 0;
    Erreur EXCEPTION;
BEGIN
    SELECT COUNT(EPIISODE.idEpisode) INTO nbEpisode
    FROM EPIISODE
    INNER JOIN PARTIE ON EPIISODE.PartieEpisode = PARTIE.idPartie
    WHERE PARTIE.idPartie = Partie;
    IF nbEpisode<=0 THEN RAISE Erreur;
    END IF;
    RETURN nbEpisode;
EXCEPTION
    WHEN Erreur THEN dbms_output.put_line('ERREUR: il n y a aucun épisode');
    RETURN nbEpisode;
END;
/
```

# Nos procédures

Cette procédure permet de connaître les relations d'un personnage selon son surnom. S'il possède aucune relation, une exception sera relevé.

```
CREATE OR REPLACE PROCEDURE RelationDe(SurnomPersonnageCherche PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesRelations is
  SELECT DISTINCT p2.NomPersonnage, p2.PrenomPersonnage, TypeRelation
  FROM PERSONNAGE p2
  INNER JOIN est_en_lien ON est_en_lien.Personnage1 = p2.idPersonnage
  INNER JOIN PERSONNAGE p1 ON est_en_lien.Personnage2 = p1.idPersonnage
  WHERE p1.SurnomPersonnage LIKE (SurnomPersonnageCherche);
BEGIN
  dbms_output.put_line('Parmi les ' || nbRelation(SurnomPersonnageCherche) || ' il y a : ');
  dbms_output.put_line('');
  FOR UneRelation in LesRelations
  LOOP
    dbms_output.put_line(UneRelation.NomPersonnage || ' ' || UneRelation.PrenomPersonnage || '. Ils sont ' || UneRelation.TypeRelation || '.');
  END LOOP;
  EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Pas de relation existante');
END RelationDe;
/
```

Cette procédure permet de connaître les relations d'un personnage selon son nom et prénom. S'il possède aucune relation, une exception sera relevé.

```
CREATE OR REPLACE PROCEDURE RelationParNomDe(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE, PrenomPersonnageCherche PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesRelations is
  SELECT DISTINCT p2.NomPersonnage, p2.PrenomPersonnage, TypeRelation
  FROM PERSONNAGE p2
  INNER JOIN est_en_lien ON est_en_lien.Personnage1 = p2.idPersonnage
  INNER JOIN PERSONNAGE p1 ON est_en_lien.Personnage2 = p1.idPersonnage
  WHERE p1.NomPersonnage LIKE (NomPersonnageCherche)
  AND p1.PrenomPersonnage LIKE (PrenomPersonnageCherche);
BEGIN
  dbms_output.put_line('Parmi les relations il y a : ');
  dbms_output.put_line('');
  FOR UneRelation in LesRelations
  LOOP
    dbms_output.put_line(UneRelation.NomPersonnage || ' ' || UneRelation.PrenomPersonnage || '. Ils sont ' || UneRelation.TypeRelation || '.');
  END LOOP;
  EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Pas de relation existante');
END RelationParNomDe;
/
```

Cette procédure permet de connaître les évènements réalisé par un personnage selon son surnom. Si le personnage n'a causé aucun évènement, une exception sera relevé.

```
CREATE OR REPLACE PROCEDURE Evenement_Cause_Par(SurnomPersonnageCherche PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesEvenement is
  SELECT DISTINCT DescriptionEvenement
  FROM EVENEMENT
  INNER JOIN cause_par ON EVENEMENT.idEvenement = cause_par.EvenementProduit
  INNER JOIN PERSONNAGE ON PERSONNAGE.idPersonnage = cause_par.PersonnageDeclencheur
  WHERE PERSONNAGE.SurnomPersonnage LIKE(SurnomPersonnageCherche);
BEGIN
  dbms_output.put_line(SurnomPersonnageCherche || ' a causé : ');
  FOR UnEvenement in LesEvenement
  LOOP
    dbms_output.put_line(UnEvenement.DescriptionEvenement);
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END Evenement_Cause_Par;
/
```

Cette procédure permet de connaître les évènements réalisé par un personnage selon son nom et prénom. Si le personnage n'a causé aucun évènement, une exception sera relevé.

```
CREATE OR REPLACE PROCEDURE Evenement_Cause_Par_Nom(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE, PrenomPersonnageCherche PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesEvenement is
  SELECT DISTINCT DescriptionEvenement
  FROM EVENEMENT
  INNER JOIN cause_par ON EVENEMENT.idEvenement = cause_par.EvenementProduit
  INNER JOIN PERSONNAGE ON PERSONNAGE.idPersonnage = cause_par.PersonnageDeclencheur
  WHERE PERSONNAGE.NomPersonnage LIKE (NomPersonnageCherche)
  AND PERSONNAGE.PrenomPersonnage LIKE (PrenomPersonnageCherche);
BEGIN
  dbms_output.put_line(NomPersonnageCherche || ' ' || PrenomPersonnageCherche || ' a causé : ');
  FOR UnEvenement in LesEvenement
  LOOP
    dbms_output.put_line(UnEvenement.DescriptionEvenement);
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END Evenement_Cause_Par_Nom;
/
```

Procédure permettant de voir quel lieux ont été visité par un personnage selon son surnom. S'il y a aucun lieu, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE A_Visite(SurnomPersonnageCherche PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesVisites is
  SELECT DISTINCT NomLieu, NomVille, NomPays
  FROM LIEU
  INNER JOIN VILLE ON LIEU.VilleLieu = VILLE.idVille
  INNER JOIN PAYS ON VILLE.VillePays= PAYS.idPays
  INNER JOIN EVENEMENT ON EVENEMENT.idEvenement = Lieu.idLieu
  INNER JOIN cause_par ON EVENEMENT.idEvenement = cause_par.EvenementProduit
  INNER JOIN PERSONNAGE ON PERSONNAGE.idPersonnage = cause_par.PersonnageDeclencheur
  INNER JOIN EPISODE ON PERSONNAGE.PremiereApparition = EPISODE.idEpisode
  WHERE PERSONNAGE.SurnomPersonnage LIKE(SurnomPersonnageCherche);
BEGIN
  dbms_output.put_line(SurnomPersonnageCherche || ' a visité : ');
  FOR UneVisite in LesVisites
  LOOP
    dbms_output.put_line(UneVisite.NomLieu || ' à ' || UneVisite.NomVille || ' dans le pays de ' || UneVisite.NomPays || '.')
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun lieu n a été visité');
END A_Visite;
/
DECLARE
```

Procédure permettant de voir quel lieux ont été visité par un personnage selon son nom et prénom. S'il y a aucun lieu, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE A_Visite_Nom(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE, PrenomPersonnageCherche PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesVisites is
  SELECT DISTINCT NomLieu, NomVille, NomPays
  FROM LIEU
  INNER JOIN VILLE ON LIEU.VilleLieu = VILLE.idVille
  INNER JOIN PAYS ON VILLE.VillePays= PAYS.idPays
  INNER JOIN EVENEMENT ON EVENEMENT.idEvenement = Lieu.idLieu
  INNER JOIN cause_par ON EVENEMENT.idEvenement = cause_par.EvenementProduit
  INNER JOIN PERSONNAGE ON PERSONNAGE.idPersonnage = cause_par.PersonnageDeclencheur
  INNER JOIN EPISODE ON PERSONNAGE.PremiereApparition = EPISODE.idEpisode
  WHERE PERSONNAGE.NomPersonnage LIKE (NomPersonnageCherche)
  AND PERSONNAGE.PrenomPersonnage LIKE (PrenomPersonnageCherche);
BEGIN
  dbms_output.put_line(NomPersonnageCherche || ' ' || PrenomPersonnageCherche || ' a visité : ');
  FOR UneVisite in LesVisites
    LOOP
      dbms_output.put_line(UneVisite.NomLieu || ' à ' || UneVisite.NomVille || ' dans le pays de ' || UneVisite.NomPays || '.');
    END LOOP;
  EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun lieu a été visité');
END A_Visite_Nom;
/
```

Procédure permettant de savoir l'âge d'un personnage selon son nom et prénom. Si la procédure trop de ligne, une exception est relevée. Ainsi que si la procédure ne relève qu'une seule ligne.

```
CREATE OR REPLACE PROCEDURE Nom_Age_De(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE, PrenomPersonnageCherche PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesAges is
  SELECT AgeIndividu
  FROM INDIVIDU
  JOIN PERSONNAGE ON PERSONNAGE.Auteur = INDIVIDU.idIndividu
  WHERE PERSONNAGE.NomPersonnage = NomPersonnageCherche
  AND PERSONNAGE.PrenomPersonnage = PrenomPersonnageCherche;
BEGIN
  dbms_output.put('L acteur de ' || NomPersonnageCherche || ' ' || PrenomPersonnageCherche || ' est âgé de ');
  FOR UnAge in LesAges
    LOOP
      dbms_output.put_line(UnAge.AgeIndividu|| ' ans.');
    END LOOP;
  EXCEPTION
    WHEN TOO_MANY_ROWS then DBMS_OUTPUT.PUT_LINE('Trop d âge trouvé');
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun âge trouvé');
END Nom_Age_De;
/
```

Procédure permettant de savoir l'âge d'un personnage selon son surnom. Si la procédure trop de ligne, une exception est relevée. Ainsi que si la procédure ne relève qu'une seule ligne.

```
CREATE OR REPLACE PROCEDURE Age_De(SurnomPersonnageCherche PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesAges is
  SELECT AgeIndividu
  FROM INDIVIDU
  JOIN PERSONNAGE ON PERSONNAGE.Auteur = INDIVIDU.idIndividu
  WHERE PERSONNAGE.SurnomPersonnage = SurnomPersonnageCherche;
BEGIN
  dbms_output.put( 'L''acteur de ' || SurnomPersonnageCherche || ' est âgé de ');
  FOR UnAge in LesAges
    LOOP
      dbms_output.put_line(UnAge.AgeIndividu|| ' ans.');
    END LOOP;
  EXCEPTION
    WHEN TOO_MANY_ROWS then DBMS_OUTPUT.PUT_LINE('Trop d âge trouvé');
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun âge trouvé');
END Age_De;
/
```

Procédure permettant de connaître le nombre d'évènement ayant lieu dans une partie S'il y a aucun évènement, une exception est relevée.

```

CREATE OR REPLACE PROCEDURE Nombre_Evenement(PartieCherche PARTIE.idPartie%TYPE)
is
Cursor LesParties is
SELECT COUNT(EVENEMENT.idEvenement) AS nbEvenement
FROM EVENEMENT
INNER JOIN EPISODE ON EPISODE.idEpisode = EVENEMENT.EpisodeEvenement
INNER JOIN PARTIE ON EPISODE.PartieEpisode = PARTIE.idPartie
WHERE PARTIE.idPartie LIKE(PartieCherche);
BEGIN
FOR UnePartie in LesParties
LOOP
    dbms_output.put_line('Il s''est déroulé ' || UnePartie.nbEvenement || ' évènement dans la partie ' || PartieCherche || '.');
END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END Nombre_Evenement;
/

```

Procédure permettant de connaître le nombre d'évènement causé par un personnage selon son nom et prénom S'il y a aucun évènement, une exception est relevée.

```

CREATE OR REPLACE PROCEDURE Nombre_Evenement_Cause_Par(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE, PrenomPersonnageCherche PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesEvenements is
SELECT COUNT(EVENEMENT.idEvenement) AS nbEvenement
FROM EVENEMENT
INNER JOIN EPISODE ON EPISODE.idEpisode = EVENEMENT.EpisodeEvenement
INNER JOIN PERSONNAGE ON EPISODE.idEpisode = PERSONNAGE.PremiereApparition
WHERE PERSONNAGE.NomPersonnage LIKE(NomPersonnageCherche)
AND PERSONNAGE.PrenomPersonnage LIKE(PrenomPersonnageCherche);
BEGIN
FOR UnEvenement in LesEvenements
LOOP
    dbms_output.put_line('Le personnage ' || NomPersonnageCherche || ' ' || PrenomPersonnageCherche || ' a causé ' || UnEvenement.nbEvenement || ' évènements.');
END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END Nombre_Evenement_Cause_Par;
/

```

Procédure permettant de connaître le nombre d'évènement causé par un personnage selon son surnom. S'il y a aucun évènement, une exception est relevée.

```

CREATE OR REPLACE PROCEDURE Nombre_Evenement_Cause_Par_Surnom(SurnomPersonnageCherche PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesEvenements is
SELECT COUNT(EVENEMENT.idEvenement) AS nbEvenement
FROM EVENEMENT
INNER JOIN EPISODE ON EPISODE.idEpisode = EVENEMENT.EpisodeEvenement
INNER JOIN PERSONNAGE ON EPISODE.idEpisode = PERSONNAGE.PremiereApparition
WHERE PERSONNAGE.SurnomPersonnage LIKE(SurnomPersonnageCherche);
BEGIN
FOR UnEvenement in LesEvenements
LOOP
    dbms_output.put_line('Le personnage ' || SurnomPersonnageCherche || ' a causé ' || UnEvenement.nbEvenement || ' évènements.');
END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END Nombre_Evenement_Cause_Par_Surnom;
/

```

Procédure permettant de savoir quels acteurs sont présents dans une partie. S'il y a aucun acteur, une exception est relevée.

```

CREATE OR REPLACE PROCEDURE ActeurPresent(PartieCherche PARTIE.idPartie%TYPE)
is
Cursor LesActeurs is
  SELECT DISTINCT INDIVIDU.NomIndividu, INDIVIDU.PrenomIndividu
  FROM PERSONNAGE
  INNER JOIN INDIVIDU ON PERSONNAGE.Auteur = INDIVIDU.idIndividu
  INNER JOIN EPISODE ON PERSONNAGE.PremiereApparition = EPISODE.idEpisode
  INNER JOIN EPISODE ON PERSONNAGE.DerniereApparition = EPISODE.idEpisode
  INNER JOIN EPISODE ON PERSONNAGE.MortPersonnage = EPISODE.idEpisode
  INNER JOIN PARTIE ON EPISODE.PartieEpisode = PARTIE.idPartie
  WHERE PARTIE.idPartie = PartieCherche
  AND PERSONNAGE.PremiereApparition is NOT NULL
  AND PERSONNAGE.DerniereApparition NOT IN (SELECT PERSONNAGE.DerniereApparition
                                              FROM PERSONNAGE
                                              INNER JOIN EPISODE ON PERSONNAGE.DerniereApparition = EPISODE.idEpisode
                                              INNER JOIN PARTIE ON PARTIE.idPartie = EPISODE.PartieEpisode
                                              WHERE PARTIE.idPartie < PartieCherche)
  AND PERSONNAGE.MortPersonnage NOT IN (SELECT PERSONNAGE.MortPersonnage
                                         FROM PERSONNAGE
                                         INNER JOIN EPISODE ON PERSONNAGE.MortPersonnage = EPISODE.idEpisode
                                         INNER JOIN PARTIE ON PARTIE.idPartie = EPISODE.PartieEpisode
                                         WHERE PARTIE.idPartie < PartieCherche);
BEGIN
  dbms_output.put_line('Les acteurs présents dans la partie ' || PartieCherche || ' sont : ');
  FOR UnActeur in LesActeurs
  LOOP
    dbms_output.put_line(UnActeur.NomIndividu || ' ' || UnActeur.PrenomIndividu );
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun acteur');
END ActeurPresent;
/

```

Procédure permettant de savoir le nombre d'évènement réalisé par un personnage dans un lieu. S'il y a aucun évènement, une exception est relevée.

```

CREATE OR REPLACE PROCEDURE NbEvenement_Cause_Par_Nom_Lieu(NomPersonnageCherche PERSONNAGE.NomPersonnage%TYPE, PrenomPersonnageCherche PERSONNAGE.PrenomPersonnage%TYPE, LieuCherche LIEU.NomLieu%TYPE)
is
Cursor LesEvenements is
  SELECT COUNT(EVENEMENT.idEvenement) AS Nb
  FROM EVENEMENT
  INNER JOIN cause_par ON EVENEMENT.idEvenement = cause_par.EvenementProduit
  INNER JOIN PERSONNAGE ON PERSONNAGE.idPersonnage = cause_par.PersonnageDeclencheur
  INNER JOIN LIEU ON LIEU.idLieu = EVENEMENT.LieuEvenement
  WHERE PERSONNAGE.NomPersonnage LIKE (NomPersonnageCherche)
  AND PERSONNAGE.PrenomPersonnage LIKE (PrenomPersonnageCherche)
  AND LIEU.NomLieu LIKE (LieuCherche);
BEGIN
  FOR UnEvenement in LesEvenements
  LOOP
    dbms_output.put_line('Le personnage ' || NomPersonnageCherche || ' ' || PrenomPersonnageCherche || ' a causé ' || UnEvenement.Nb || ' événements à ' || LieuCherche || '.' );
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END NbEvenement_Cause_Par_Nom_Lieu;
/

```

Procédure permettant de savoir quel réalisateur a réalisé une partie. S'il y a aucun réalisateur, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Realisateur_De_Partie(PartieCherche PARTIE.idPartie%TYPE)
is
Cursor LesRealisateurs is
    SELECT DISTINCT NomIndividu, PrenomIndividu
    FROM INDIVIDU
    INNER JOIN EPISODE ON EPISODE.Réalisateur = INDIVIDU.idIndividu
    INNER JOIN PARTIE ON EPISODE.PartieEpisode = PARTIE.idPartie
    WHERE PartieEpisode = PartieCherche
    AND RoleIndividu = 'Réalisateur';
BEGIN
    dbms_output.put_line('Les réalisateurs de la partie ' || PartieCherche || ' sont : ');
    FOR UnRealisateur in LesRealisateurs
    LOOP
        dbms_output.put_line(UnRealisateur.NomIndividu || ' ' || UnRealisateur.PrenomIndividu);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun réalisateur');
END Realisateur_De_Partie;
/
```

Procédure permettant de savoir quelle est la relation entre deux personnages selon leurs surnoms. S'il y a aucune relation, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Relation_Entre(Surnom1 PERSONNAGE.SurnomPersonnage%TYPE, Surnom2 PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesRelations is
    SELECT TypeRelation
    FROM PERSONNAGE pl
    JOIN est_en_lien ON pl.idPersonnage = est_en_lien.Personnage1
    JOIN PERSONNAGE p2 ON p2.idPersonnage = est_en_lien.Personnage2
    WHERE pl.SurnomPersonnage = Surnom1
    AND p2.SurnomPersonnage = Surnom2;
BEGIN
    FOR UneRelation in LesRelations
    LOOP
        dbms_output.put_line('La relation entre ' || Surnom1 || ' et ' || Surnom2 || ' est de type ' || UneRelation.TypeRelation || '.');
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Ils n ont aucune relation');
END Relation_Entre;
/
```

Procédure permettant de savoir quelle est la relation entre deux personnages selon leurs noms et prénoms. S'il y a aucune relation, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Relation_Entre_Nom(Nom1 PERSONNAGE.NomPersonnage%TYPE, Prenom1 PERSONNAGE.PrenomPersonnage%TYPE, Nom2 PERSONNAGE.NomPersonnage%TYPE, Prenom2 PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesRelations is
    SELECT TypeRelation
    FROM PERSONNAGE pl
    JOIN est_en_lien ON pl.idPersonnage = est_en_lien.Personnage1
    JOIN PERSONNAGE p2 ON p2.idPersonnage = est_en_lien.Personnage2
    WHERE pl.NomPersonnage = Nom1
    AND pl.PrenomPersonnage = Prenom1
    AND p2.NomPersonnage = Nom2
    AND p2.PrenomPersonnage = Prenom2;
BEGIN
    FOR UneRelation in LesRelations
    LOOP
        dbms_output.put_line('La relation entre ' || Nom1 || ' ' || Prenom1 || ' et ' || Nom2 || ' ' || Prenom2 || ' est de type ' || UneRelation.TypeRelation || '.');
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Ils n ont aucune relation');
END Relation_Entre_Nom;
/
```

## Procédure permettant de savoir quand est apparût un personnage selon son surnom.

```
CREATE OR REPLACE PROCEDURE Apparition_Partie_Surnom(SurnomCherche PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesParties is
  SELECT DISTINCT idPartie
  FROM Partie
  INNER JOIN EPISODE ON PARTIE.idPartie = EPISODE.PartieEpisode
  INNER JOIN PERSONNAGE ON EPISODE.idEpisode = PERSONNAGE.PremiereApparition
  WHERE PERSONNAGE.SurnomPersonnage = SurnomCherche;
BEGIN
  FOR UnePartie in LesParties
  LOOP
    dbms_output.put_line('Le personnage ' || SurnomCherche || ' est apparût dans la partie numéro ' || UnePartie.idPartie || '.');
  END LOOP;
END Apparition_Partie_Surnom;
/
```

## Procédure permettant de savoir quand est apparût un personnage selon son nom et prénom.

```
CREATE OR REPLACE PROCEDURE Apparition_Partie_Nom(Nom PERSONNAGE.NomPersonnage%TYPE, Prenom PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesParties is
  SELECT DISTINCT idPartie
  FROM Partie
  INNER JOIN EPISODE ON PARTIE.idPartie = EPISODE.PartieEpisode
  INNER JOIN PERSONNAGE ON EPISODE.idEpisode = PERSONNAGE.PremiereApparition
  WHERE PERSONNAGE.PrenomPersonnage = Prenom
  AND PERSONNAGE.NomPersonnage = Nom;
BEGIN
  FOR UnePartie in LesParties
  LOOP
    dbms_output.put_line('Le personnage ' || Nom || ' ' || Prenom || ' est apparût dans la partie numéro ' || UnePartie.idPartie || '.');
  END LOOP;
END Apparition_Partie_Nom;
/
```

## Procédure permettant de savoir dans quel épisode est mort un personnage selon son surnom. Si le personnage n'est pas mort, un exception est relevée.

```
CREATE OR REPLACE PROCEDURE Mort_A(Surnom PERSONNAGE.SurnomPersonnage%TYPE)
is
Cursor LesEpisodes is
  SELECT NumeroEpisode, PartieEpisode
  FROM EPISODE
  JOIN PERSONNAGE ON PERSONNAGE.MortPersonnage = EPISODE.idEpisode
  WHERE PERSONNAGE.SurnomPersonnage = Surnom;
BEGIN
  FOR UnEpisode in LesEpisodes
  LOOP
    dbms_output.put_line('Le personnage ' || Surnom || ' est mort dans l épisode ' || UnEpisode.NumeroEpisode || ' de la parti numéro ' || UnEpisode.PartieEpisode || '.');
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Le personnage n est pas mort.');
END Mort_A;
/
```

## Procédure permettant de savoir dans quel épisode est mort un personnage selon son nom et prénom. Si le personnage n'est pas mort, un exception est relevée.

```
CREATE OR REPLACE PROCEDURE Mort_A(Nom PERSONNAGE.NomPersonnage%TYPE, Prenom PERSONNAGE.PrenomPersonnage%TYPE)
is
Cursor LesEpisodes is
  SELECT NumeroEpisode, PartieEpisode
  FROM EPISODE
  JOIN PERSONNAGE ON PERSONNAGE.MortPersonnage = EPISODE.idEpisode
  WHERE PERSONNAGE.NomPersonnage = Nom
  AND PERSONNAGE.PrenomPersonnage = Prenom;
BEGIN
  FOR UnEpisode in LesEpisodes
  LOOP
    dbms_output.put_line('Le personnage ' || Nom || ' ' || Prenom || ' est mort dans l épisode ' || UnEpisode.NumeroEpisode || ' de la parti numéro ' || UnEpisode.PartieEpisode || '.');
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Le personnage n est pas mort.');
END Mort_A;
/
```

Procédure permettant de savoir quels évènements se sont déroulés dans un lieu. S'il y a aucun évènement, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Evenement_Lieu(Nom LIEU.NomLieu%TYPE)
is
Cursor LesEvenements is
    SELECT DISTINCT EVENEMENT.DescriptionEvenement
    FROM EVENEMENT
    JOIN LIEU ON EVENEMENT.LieuEvenement = LIEU.idLieu
    WHERE LIEU.NomLieu LIKE (Nom);
BEGIN
    dbms_output.put_line('Les évènements ayant eu lieu à '|| Nom || ' sont : ');
    FOR UnEvenement in LesEvenements
    LOOP
        dbms_output.put_line(UnEvenement.DescriptionEvenement);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun évènement');
END Evenement_Lieu;
/
```

Procédure permettant de savoir qui a réalisé un épisode. S'il y a aucun réalisateur, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Episode_Par(Nom INDIVIDU.NomIndividu%TYPE, Prenom INDIVIDU.PrenomIndividu%TYPE)
is
Cursor LesEpisodes is
    SELECT DISTINCT EPISODE.NumeroEpisode, EPISODE.PartieEpisode, EPISODE.TitreEpisodeVO, EPISODE.TitreEpisodeVF
    FROM EPISODE
    JOIN INDIVIDU ON EPISODE.Réalisateur = INDIVIDU.idIndividu
    WHERE INDIVIDU.NomIndividu LIKE (Nom)
    AND INDIVIDU.PrenomIndividu LIKE (Prenom);
BEGIN
    dbms_output.put_line('Le réalisateur ' || Nom || ' ' || Prenom || ' a réalisé les épisodes suivants : ');
    FOR UnEpisode in LesEpisodes
    LOOP
        dbms_output.put_line(UnEpisode.NumeroEpisode || ' de la partie ' || UnEpisode.PartieEpisode || ' dont le titre original est ' || UnEpisode.TitreEpisodeVO || '. La traduction est' || UnEpisode.TitreEpisodeVF);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucun épisode');
END Episode_Par;
/
```

Procédure permettant de savoir qui ont une relations d'un certain type. S'il y a aucune relation de ce type, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Relation_De_Type(Type est_en_lien.TypeRelation%TYPE)
is
Cursor LesPersonnages is
    SELECT DISTINCT PERSONNAGE.NomPersonnage, PERSONNAGE.PrenomPersonnage
    FROM PERSONNAGE
    INNER JOIN est_en_lien ON PERSONNAGE.idPersonnage = est_en_lien.Personnage1
    WHERE est_en_lien.TypeRelation = Type;
BEGIN
    dbms_output.put_line('Les personnages dont la relation est ' || Type || ' sont : ');
    FOR UnPersonnage in LesPersonnages
    LOOP
        dbms_output.put_line(UnPersonnage.NomPersonnage || ' ' || UnPersonnage.PrenomPersonnage);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Aucune relation de ce type');
END Relation_De_Type;
/
```

Procédure permettant de savoir quels évènements se sont déroulés dans une ville. S'il y a aucun évènement, une exception est relevée.

```
CREATE OR REPLACE PROCEDURE Evenement_Ville(NomV Ville.NomVille%TYPE)
is
Cursor LesEvenements is
  SELECT EV.DescriptionEvenement
  FROM EVENEMENT EV
  JOIN LIEU L ON EV.LieuEvenement = L.idLieu
  JOIN VILLE V ON L.VilleLieu = V.idVille
  JOIN EPISODE E ON E.idEpisode = EV.EpisodeEvenement
  WHERE V.NomVille = NomV
  ORDER BY E.idEpisode;
BEGIN
  dbms_output.put_line('Les evenements qui ont lieu à '||NomV|| ' sont : ');
  FOR UnEvenement in LesEvenements
  LOOP
    dbms_output.put_line(' - ' || UnEvenement.DescriptionEvenement);
  END LOOP;
END Evenement_Ville;
/
```

# Nos réponses aux questions

```
-- QUESTION 1 : Quel est l'age de l'actrice de Lisbonne ? --
/*Selon le surnom*/
DECLARE
BEGIN
    Age_De('Lisbonne'); --Voir Procedure n°8--
END;
/
/*Selon le nom*/
DECLARE
BEGIN
    Nom_Age_De('Murillo', 'Raquel'); --Voir Procedure n°7--
END;
/
-- QUESTION 2 : Quels sont les lieux que « NomPersonnage » a visité ?
/*Selon le nom*/
DECLARE
BEGIN
    A_Visite_Nom('Marquina', 'Sergio'); --Voir Procedure n°6--
END;
/
/*Selon le surnom*/
DECLARE
BEGIN
    A_Visite('le Professeur'); --Voir Procedure n°5--
END;
/
```

```

-- QUESTION 3 : Combien d'évènement se déroule dans la partie 1 ? --
DECLARE
BEGIN
    Nombre_Evenement(1); --Voir Procedure n°9--
END;
/

-- QUESTION 4 : Combien d'évènement à causé le personnage de Raquel Murillo durant l'épisode de son apparition ? --
/*Selon le nom et prénom*/
DECLARE
BEGIN
    Nombre_Evenement_Cause_Par('Murillo', 'Raquel'); -- Voir Procedure n°10--
END;
/
/*Selon le surnom*/
DECLARE
BEGIN
    Nombre_Evenement_Cause_Par_Surnom('Lisbonne'); -- Voir Procedure n°11--
END;
/

-- QUESTION 5 : Quels sont les évènements causé par le personnage de Raquel Murillo ? --
/* Selon le surnom*/
DECLARE
BEGIN
    Evenement_Cause_Par('Lisbonne'); -- Voir Procedure n°4--
END;
/
/* Selon le nom et prénom*/
DECLARE
BEGIN
    Evenement_Cause_Par_Nom('Murillo', 'Raquel'); -- Voir Procedure n°3--
END;
/

-- QUESTION 6 : Quels sont les personnages qui ne sont pas réapparu ? --
SELECT * FROM PersonnageDisparu ORDER BY EpisodeDisparu ASC; --Voir vue n°4--

-- QUESTION 7 : Quels sont les acteurs qui sont présent dans la partie 2 ? --
DECLARE
BEGIN
    ActeurPresent(2);--Voir Procédure n°12--
END;
/

-- QUESTION 8 : Combien d'évènement a causé le personnage Murillo à la Banque d'Espagne ? --
BEGIN
    NbEvenement_Cause_Par_Nom_Lieu('Murillo', 'Raquel', 'Banque d Espagne');--Voir Procédure n°13--
END;
/

-- QUESTION 9 : Quelle est la liste des réalisateurs de la partie 3 ? --
DECLARE
BEGIN
    Realisateur_De_Partie(3);--Voir Procédure n°14--
END;
/

```

-- QUESTION 10 : Quel est la relation entre les personnages Tokyo et Rio ? --

```
DECLARE
BEGIN
    Relation_Entre('Tokyo', 'Rio');--Voir Procédure n°15--
END;
/
```

-- QUESTION 11 : Dans quel partie apparaît le personnage Tokyo ? --

```
DECLARE
BEGIN
    Apparition_Partie_Surnom('Tokyo');--Voir Procédure n°17--
END;
/
```

-- QUESTION 12 : Combien de personnage sont mort dans la partie 2 ? --

```
DECLARE
BEGIN
    DBMS_OUTPUT.PUT_LINE(nbMort(2));--Voir Fonction n°6--
END;
/
```

-- QUESTION 13 : Quels sont les évènements causé dans le lieu à la Banque? --

```
EXECUTE Evenement_Ville('Madrid') ;--Voir Procédure n°X--
```

-- QUESTION 14 : Quels sont les personnages reliés au personnage « NomPersonnage » ? --

```
DECLARE
BEGIN
    RelationDe('Rio');--Voir Procédure n°1--
END;
/
```

-- QUESTION 15 : Dans quel épisode le personnage Tokyo est mort ? --

```
DECLARE
BEGIN
    Mort_A('Tokyo');--Voir Procédure n°19--
END;
/
```

-- QUESTION 16 : Quels sont les évènements causé dans la ville Madrid ? --

```
DECLARE
BEGIN
    Evenement_Lieu('Banque d Espagne') --Voir Procédure n°21--
END;
```

-- QUESTION 17 : Combien y a-t-il d'épisode dans la partie 2 ? --

```
DECLARE
BEGIN
    DBMS_OUTPUT.PUT_LINE(nbEpisode_Partie(2)); -- Voir fonction n°7--
END;
/
```

```
-- QUESTION 18 : Quels sont les épisodes que le réalisateur Quintas Javier a réalisé ? --
DECLARE
BEGIN
    Episode_Par('Quintas', 'Javier'); -- Voir Procédure n°22--
END;
/

-- QUESTION 19 : Quels sont les personnages en relation de type « TypeRelation » ? --
DECLARE
BEGIN
    Relation_De_Type('Coéquipier'); --Voir Procédure n°23--
END;
/

-- QUESTION 20: Combien d'épisode a réalisé Rodiguo Alex ?
DECLARE
BEGIN
    DBMS_OUTPUT.PUT_LINE(nbEpisode('Rodrigo','Alex'))); -- Voir fonction numéro 1--
END;
/
```

# Nos créations de tables

```
CREATE TABLE INDIVIDU(
    idIndividu NUMBER(5),
    NomIndividu VARCHAR(50) NOT NULL,
    PrenomIndividu VARCHAR(50),
    AgeIndividu NUMBER(2),
    RoleIndividu VARCHAR(50) NOT NULL,
    CONSTRAINT cleINDIVIDU PRIMARY KEY(idIndividu),
    CONSTRAINT verifAgeIndividu CHECK((AgeIndividu>0) OR (AgeIndividu is NULL))
);

CREATE TABLE PARTIE(
    idPartie NUMBER(5),
    DateDiffusion DATE NOT NULL,
    CONSTRAINT clePARTIE PRIMARY KEY(idPartie)
);

CREATE TABLE EPISODE(
    idEpisode NUMBER(5),
    NumeroEpisode NUMBER(5) NOT NULL,
    TitreEpisodeVO VARCHAR(50),
    TitreEpisodeVF VARCHAR(50),
    Realisateur NUMBER(5),
    PartieEpisode NUMBER(5) NOT NULL,
    CONSTRAINT cleEPISODE PRIMARY KEY(idEpisode),
    CONSTRAINT cleREALISATEUR FOREIGN KEY(Realisateur) REFERENCES INDIVIDU(idIndividu),
    CONSTRAINT clePARTIEEPISODE FOREIGN KEY(PartieEpisode) REFERENCES PARTIE(idPartie),
    CONSTRAINT verifNumeroEpisode CHECK(NumeroEpisode>0)
);
```

```

CREATE TABLE PAYS (
    idPays NUMBER(5),
    NomPays VARCHAR(50) NOT NULL,
    CONSTRAINT clePAYS PRIMARY KEY(idPays)
);

CREATE TABLE PERSONNAGE (
    idPersonnage NUMBER(5),
    NomPersonnage VARCHAR(50),
    PrenomPersonnage VARCHAR(50),
    SurnomPersonnage VARCHAR(50),
    PremiereApparition NUMBER(5) NOT NULL,
    DerniereApparition NUMBER(5) NOT NULL,
    MortPersonnage NUMBER(5),
    Acteur NUMBER(5) NOT NULL,
    CONSTRAINT clePERSONNAGE PRIMARY KEY(idPersonnage),
    CONSTRAINT clePREMIEREAPPARITION FOREIGN KEY(PremiereApparition) REFERENCES EPISODE(idEpisode),
    CONSTRAINT cleDERNIEREAPPARTION FOREIGN KEY(DerniereApparition) REFERENCES EPISODE(idEpisode),
    CONSTRAINT cleMORTPERSONNAGE FOREIGN KEY(MortPersonnage) REFERENCES EPISODE(idEpisode),
    CONSTRAINT cleACTEUR FOREIGN KEY(Acteur) REFERENCES INDIVIDU(idIndividu)
);

CREATE TABLE VILLE(
    idVille NUMBER(5),
    NomVille VARCHAR(50) NOT NULL,
    VillePays NUMBER(5) NOT NULL,
    CONSTRAINT cleVILLE PRIMARY KEY(idVille),
    CONSTRAINT cleVILLEPAYS FOREIGN KEY(VillePays) REFERENCES PAYS(idPays)
);

CREATE TABLE LIEU(
    idLieu NUMBER(5),
    NomLieu VARCHAR(50) NOT NULL,
    VilleLieu NUMBER(5) NOT NULL,
    CONSTRAINT cleLIEU PRIMARY KEY(idLieu),
    CONSTRAINT cleVILLELIEU FOREIGN KEY(VilleLieu) REFERENCES VILLE(idVille)
);

CREATE TABLE EVENEMENT(
    idEvenement NUMBER(5),
    DescriptionEvenement VARCHAR(150) NOT NULL,
    EpisodeEvenement NUMBER(5) NOT NULL,
    LieuEvenement NUMBER(5),
    CONSTRAINT cleEVENEMENT PRIMARY KEY(idEvenement),
    CONSTRAINT cleEPISODEEVENEMENT FOREIGN KEY(EpisodeEvenement) REFERENCES EPISODE(idEpisode),
    CONSTRAINT cleLIEUEVENEMENT FOREIGN KEY(LieuEvenement) REFERENCES LIEU(idLieu)
);

CREATE TABLE cause_par(
    PersonnageDeclencheur NUMBER(5),
    EvenementProduit NUMBER(5),
    CONSTRAINT cleCAUSE_PAR PRIMARY KEY(PersonnageDeclencheur, EvenementProduit),
    CONSTRAINT clePERSONNAGEDECLENCHEUR FOREIGN KEY(PersonnageDeclencheur) REFERENCES PERSONNAGE(idPersonnage),
    CONSTRAINT cleEVENEMENTPRODUIT FOREIGN KEY(EvenementProduit) REFERENCES EVENEMENT(idEvenement)
);

```

```
CREATE TABLE est_en_lien(
    idLien NUMBER(5),
    Personnage1 NUMBER(5) NOT NULL,
    Personnage2 NUMBER(5) NOT NULL,
    TypeRelation VARCHAR(50) NOT NULL,
    CONSTRAINT cleEST_EN_LIEN PRIMARY KEY(idLien),
    CONSTRAINT clePERSONNAGE1 FOREIGN KEY(Personnage1) REFERENCES PERSONNAGE(idPersonnage),
    CONSTRAINT clePERSONNAGE2 FOREIGN KEY(Personnage2) REFERENCES PERSONNAGE(idPersonnage)
);
```

# Nos insertions de données

```
--Insertion des données dans la table PARTIE--
```

```
INSERT INTO PARTIE VALUES(1, '20/12/2017');  
INSERT INTO PARTIE VALUES(2, '06/05/2018');  
INSERT INTO PARTIE VALUES(3, '19/07/2019');  
INSERT INTO PARTIE VALUES(4, '03/05/2020');  
INSERT INTO PARTIE VALUES(5, '03/12/2021');
```

```
--Insertion des données dans la table INDIVIDU--
```

```
INSERT INTO INDIVIDU VALUES(1, 'Morte', 'Alvaro', 48, 'Acteur');  
INSERT INTO INDIVIDU VALUES(2, 'Corbero', 'Úrsula', 33, 'Acteur');  
INSERT INTO INDIVIDU VALUES(3, 'Ituño', 'Itziar', 48, 'Acteur');  
INSERT INTO INDIVIDU VALUES(4, 'Alonso', 'Pedro', 51, 'Acteur');  
INSERT INTO INDIVIDU VALUES(5, 'Flores', 'Alba', 36, 'Acteur');  
INSERT INTO INDIVIDU VALUES(6, 'Harràn', 'Miguel', 27, 'Acteur');  
INSERT INTO INDIVIDU VALUES(7, 'Lorrente', 'Jaime', 31, 'Acteur');  
INSERT INTO INDIVIDU VALUES(8, 'Tous', 'Paco', 59, 'Acteur');  
INSERT INTO INDIVIDU VALUES(9, 'Peric', 'Darko', 46, 'Acteur');  
INSERT INTO INDIVIDU VALUES(10, 'Acebo', 'Esther', 40, 'Acteur');  
INSERT INTO INDIVIDU VALUES(11, 'Garcia', 'Roberto', 49, 'Acteur');  
INSERT INTO INDIVIDU VALUES(12, 'Arce', 'Enrique', 50, 'Acteur');  
INSERT INTO INDIVIDU VALUES(13, 'Soto', 'Fernando', 54, 'Acteur');  
INSERT INTO INDIVIDU VALUES(14, 'Pedraza', 'Maria', 27, 'Acteur');  
INSERT INTO INDIVIDU VALUES(15, 'Manver', 'Kiti', 70, 'Acteur');  
INSERT INTO INDIVIDU VALUES(16, 'de la Selva', 'Rodrigo', 47, 'Acteur');  
INSERT INTO INDIVIDU VALUES(17, 'Keuchkerian', 'Hovik', 50, 'Acteur');  
INSERT INTO INDIVIDU VALUES(18, 'Perros', 'Luka', 46, 'Acteur');  
INSERT INTO INDIVIDU VALUES(19, 'Cuesta', 'Belén', 39, 'Acteur');  
INSERT INTO INDIVIDU VALUES(20, 'Cayo', 'Fernando', 55, 'Acteur');  
INSERT INTO INDIVIDU VALUES(21, 'Nimri', 'Najwa', 51, 'Acteur');  
INSERT INTO INDIVIDU VALUES(22, 'Colmenar', 'Jesús', NULL, 'Réalisateur');  
INSERT INTO INDIVIDU VALUES(23, 'Serra', 'Koldo', 48, 'Réalisateur');  
INSERT INTO INDIVIDU VALUES(24, 'Rodrigo', 'Álex', 60, 'Réalisateur');
```

```

INSERT INTO INDIVIDU VALUES(24, 'Rodrigo', 'Álex', 60, 'Réalisateur');
INSERT INTO INDIVIDU VALUES(25, 'Quintas', 'Javier', 35, 'Réalisateur');

--Insertion des données dans la table EPISODE--
INSERT INTO EPISODE VALUES (1, 1, NULL, 'Episode 1', NULL, 1);
INSERT INTO EPISODE VALUES (2, 2, NULL, 'Episode 2', NULL, 1);
INSERT INTO EPISODE VALUES (3, 3, NULL, 'Episode 3', NULL, 1);
INSERT INTO EPISODE VALUES (4, 4, NULL, 'Episode 4', NULL, 1);
INSERT INTO EPISODE VALUES (5, 5, NULL, 'Episode 5', NULL, 1);
INSERT INTO EPISODE VALUES (6, 6, NULL, 'Episode 6', NULL, 1);
INSERT INTO EPISODE VALUES (7, 7, NULL, 'Episode 7', NULL, 1);
INSERT INTO EPISODE VALUES (8, 8, NULL, 'Episode 8', NULL, 1);
INSERT INTO EPISODE VALUES (9, 9, NULL, 'Episode 9', NULL, 1);
INSERT INTO EPISODE VALUES (10, 10, NULL, 'Episode 10', NULL, 1);
INSERT INTO EPISODE VALUES (11, 11, NULL, 'Episode 11', NULL, 1);
INSERT INTO EPISODE VALUES (12, 12, NULL, 'Episode 12', NULL, 1);
INSERT INTO EPISODE VALUES (13, 13, NULL, 'Episode 13', NULL, 1);
INSERT INTO EPISODE VALUES (14, 1, NULL, 'Episode 14', NULL, 2);
INSERT INTO EPISODE VALUES (15, 2, NULL, 'Episode 15', NULL, 2);
INSERT INTO EPISODE VALUES (16, 3, NULL, 'Episode 16', NULL, 2);
INSERT INTO EPISODE VALUES (17, 4, NULL, 'Episode 17', NULL, 2);
INSERT INTO EPISODE VALUES (18, 5, NULL, 'Episode 18', NULL, 2);
INSERT INTO EPISODE VALUES (19, 6, NULL, 'Episode 19', NULL, 2);
INSERT INTO EPISODE VALUES (20, 7, NULL, 'Episode 20', NULL, 2);
INSERT INTO EPISODE VALUES (21, 8, NULL, 'Episode 21', NULL, 2);
INSERT INTO EPISODE VALUES (22, 9, NULL, 'Episode 22', NULL, 2);
INSERT INTO EPISODE VALUES (23, 1, 'Hemos vuelto', 'On est de retour', 22, 3);
INSERT INTO EPISODE VALUES (24, 2, 'Aikido', 'Aikido', 22, 3);
INSERT INTO EPISODE VALUES (25, 3, '48 metros bajo el suelo', '48 mètres sous terre', 22, 3);
INSERT INTO EPISODE VALUES (26, 4, 'La hora del delfín', 'L Heure du dauphin', 23, 3);
INSERT INTO EPISODE VALUES (27, 5, 'Bum, bum, ciao', 'Boum, boum, ciao', 23, 3);
INSERT INTO EPISODE VALUES (28, 6, 'Todo pareció insignificante', 'Plus rien avait d importance', 24, 3);
INSERT INTO EPISODE VALUES (29, 7, 'Pequeñas vacaciones', 'Petites Vacances', 24, 3);
INSERT INTO EPISODE VALUES (30, 8, 'La deriva', 'La Dérive', 22, 3);
INSERT INTO EPISODE VALUES (31, 1, 'Game Over', 'Game Over', 23, 4);
INSERT INTO EPISODE VALUES (32, 2, 'La boda de Berlin', 'Le Mariage de Berlin', 25, 4);
INSERT INTO EPISODE VALUES (33, 3, 'Lección de anatomía', 'La Leçon d anatomie', 24, 4);

INSERT INTO EPISODE VALUES (34, 4, 'Suspiros de España', 'Paso-doble', 22, 4);
INSERT INTO EPISODE VALUES (35, 5, '5 minutos antes', 'Cinq minutes plus tôt', 24, 4);
INSERT INTO EPISODE VALUES (36, 6, 'KO tecnico', 'KO technique', 25, 4);
INSERT INTO EPISODE VALUES (37, 7, 'Tumbar la carpa', 'Frapper le QG', 23, 4);
INSERT INTO EPISODE VALUES (38, 8, 'Plan Paris', 'Le Plan Paris', 22, 4);
INSERT INTO EPISODE VALUES (39, 1, 'El final del camino', 'Au bout du rouleau', NULL, 5);
INSERT INTO EPISODE VALUES (40, 2, '¿Crees en la reencarnación?', 'Tu crois en la réincarnation ?', NULL, 5);
INSERT INTO EPISODE VALUES (41, 3, 'El espectáculo de la vida', 'Bienvenue au spectacle de la vie', NULL, 5);
INSERT INTO EPISODE VALUES (42, 4, 'Tu sitio en el cielo', 'Ta place au paradis', NULL, 5);
INSERT INTO EPISODE VALUES (43, 5, 'Vivir muchas vidas', 'Vis tes vies', NULL, 5);
INSERT INTO EPISODE VALUES (44, 6, 'Válvulas de escape', 'Trop de pression', NULL, 5);
INSERT INTO EPISODE VALUES (45, 7, 'Ciencia ilusionada', 'Science optimiste', NULL, 5);
INSERT INTO EPISODE VALUES (46, 8, 'La teoría de la elegancia', 'La Théorie de l élégance', NULL, 5);
INSERT INTO EPISODE VALUES (47, 9, 'Lo que se habla en la cama', 'Confidences sur l oreiller', NULL, 5);
INSERT INTO EPISODE VALUES (48, 10, 'Una tradición familiar', 'Une tradition familiale', NULL, 5);

--Insertion des données dans la table PAYS--
INSERT INTO PAYS (idPays, NomPays) VALUES (1, 'Espagne');
INSERT INTO PAYS (idPays, NomPays) VALUES (2, 'Panama');
INSERT INTO PAYS (idPays, NomPays) VALUES (3, 'Italie');
INSERT INTO PAYS (idPays, NomPays) VALUES (4, 'Indonésie');
INSERT INTO PAYS (idPays, NomPays) VALUES (5, 'Argentine');
INSERT INTO PAYS (idPays, NomPays) VALUES (6, 'Philippines');
INSERT INTO PAYS (idPays, NomPays) VALUES (7, 'Thailande');

```

```
--Insertion des données dans la table VILLE--
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (1, 'Guna Yala', 2);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (2, 'Madrid', 1);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (3, 'Florence', 3);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (4, 'Java', 4);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (5, 'La Pampa', 5);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (6, 'Tolède', 1);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (7, 'Palawan', 6);
INSERT INTO VILLE (idVille, NomVille, VillePays) VALUES (8, 'Pattaya', 7);

-- Insertions des données dans la table PERSONNAGE--
INSERT INTO PERSONNAGE VALUES (1, 'Marquina', 'Sergio', 'le Professeur', 1, 47, null, 1);
INSERT INTO PERSONNAGE VALUES (2, 'Olliveira', 'Silene', 'Tokyo', 1, 48, 47, 2);
INSERT INTO PERSONNAGE VALUES (3, 'Murillo', 'Raquel', 'Lisbonne', 1, 48, NULL, 3);
INSERT INTO PERSONNAGE VALUES (4, 'De Fonollosa', 'Andrés', 'Berlin', 1, 47, 20, 4);
INSERT INTO PERSONNAGE VALUES (5, 'Jimenez', 'Agata', 'Nairobi', 1, 38, 38, 5);
INSERT INTO PERSONNAGE VALUES (6, 'Cortés', 'Anibal', 'Rio', 1, 48, NULL, 6);
INSERT INTO PERSONNAGE VALUES (7, 'Ramos', 'Daniel', 'Denver', 1, 48, NULL, 7);
INSERT INTO PERSONNAGE VALUES (8, 'Ramos', 'Agustin', 'Moscou', 1, 22, 19, 8);
INSERT INTO PERSONNAGE VALUES (9, 'Dragic', 'Mirko', 'Helsinki', 1, 48, NULL, 9);
INSERT INTO PERSONNAGE VALUES (10, 'Gaztambide', 'Monica', 'Stockholm', 1, 48, NULL, 10);
INSERT INTO PERSONNAGE VALUES (11, 'Mostovoi', 'Dimitri', 'Oslo', 1, 6, 13, 11);
INSERT INTO PERSONNAGE VALUES (12, 'Roman', 'Arturo', 'Arturito', 1, 20, NULL, 12);
INSERT INTO PERSONNAGE VALUES (13, 'Berote', 'Martin', 'Palermme', 23, 48, NULL, 16);
INSERT INTO PERSONNAGE VALUES (14, 'Lopez', 'Santiago', 'Bogota', 23, 48, NULL, 17);
INSERT INTO PERSONNAGE VALUES (15, 'René', 'Yakov', 'Marseille', 23, 48, NULL, 18);
INSERT INTO PERSONNAGE VALUES (16, NULL, 'Julia', 'Manille', 23, 48, NULL, 19);
INSERT INTO PERSONNAGE VALUES (17, 'Rubio', 'Angel', NULL, 23, 48, NULL, 13);
INSERT INTO PERSONNAGE VALUES (18, 'Parker', 'Alison', NULL, 31, 48, NULL, 14);
INSERT INTO PERSONNAGE VALUES (19, 'Murillo', 'Marivi', NULL, 1, 30, NULL, 15);
INSERT INTO PERSONNAGE VALUES (20, 'Tamayo', 'Luis', NULL, 15, 48, NULL, 20);
INSERT INTO PERSONNAGE VALUES (21, 'Sierra', 'Alicia', NULL, 15, 48, NULL, 21);

--Insertion des données dans la table LIEU--
INSERT INTO LIEU VALUES (1, 'Archipel de Guna Yala', 1);
INSERT INTO LIEU VALUES (2, 'Banque d Espagne', 2);
INSERT INTO LIEU VALUES (3, 'Café "Le Hanoï"', 2);
INSERT INTO LIEU VALUES (4, 'Casse de voitures', 2);
INSERT INTO LIEU VALUES (5, 'Fabrique de la Monnaie', 2);
INSERT INTO LIEU VALUES (6, 'Florence', 3);
INSERT INTO LIEU VALUES (7, 'Java', 4);
INSERT INTO LIEU VALUES (8, 'La Pampa', 5);
INSERT INTO LIEU VALUES (9, 'Madrid', 2);
INSERT INTO LIEU VALUES (10, 'Maison de Raquel', 2);
INSERT INTO LIEU VALUES (11, 'Maison de Toledé', 6);
INSERT INTO LIEU VALUES (12, 'Monastere', 3);
INSERT INTO LIEU VALUES (13, 'Palawan', 7);
INSERT INTO LIEU VALUES (14, 'Tente de la police 1', 2);
INSERT INTO LIEU VALUES (15, 'Tente de la police 2', 2);
INSERT INTO LIEU VALUES (16, 'Entrepot', 2);
INSERT INTO LIEU VALUES (17, 'Camping-car du professeur', 2);
INSERT INTO LIEU VALUES (18, 'Déervoir d orage', 2);
```

```
--Insertion des données dans la table EVENEMENT--
INSERT INTO EVENEMENT VALUES (1, 'Capture de Rio.', 23, 1);
INSERT INTO EVENEMENT VALUES (2, 'Mort de Nairobi.', 38, 2);
INSERT INTO EVENEMENT VALUES (3, 'Mort de Tokyo.', 41, 2);
INSERT INTO EVENEMENT VALUES (4, 'Début du bracage.', 24, 2);
INSERT INTO EVENEMENT VALUES (5, 'Fin du bracage.', 48, 2);
INSERT INTO EVENEMENT VALUES (6, 'Rencontre Professeur et Raquel.', 2, 3);
INSERT INTO EVENEMENT VALUES (7, 'Raquel menace le professeur avec son arme et exige des explications.', 11, 3);
INSERT INTO EVENEMENT VALUES (8, 'Raquel decouvre la reel identité du professeur.', 18, 3);
INSERT INTO EVENEMENT VALUES (9, 'Le professeur se deguise en SDF pour echapper a la police.', 7, 4);
INSERT INTO EVENEMENT VALUES (10, 'Le professeur detruit les preuves dans la voiture.', 7, 4);
INSERT INTO EVENEMENT VALUES (11, 'Debut du bracage.', 1, 5);
INSERT INTO EVENEMENT VALUES (12, 'Mort de Berlin', 22, 5);
INSERT INTO EVENEMENT VALUES (13, 'Mort de Moscou', 19, 5);
INSERT INTO EVENEMENT VALUES (14, 'Mort d Oslo', 14, 5);
INSERT INTO EVENEMENT VALUES (15, 'Fin bracage.', 22, 5);
INSERT INTO EVENEMENT VALUES (16, 'Flashback. Sergio, le Professeur, qui rejoint son frère Andres, Berlin, à Florence en Italie.', 23, 6);
INSERT INTO EVENEMENT VALUES (17, 'Andres révèle à son frère qu il est atteint par la même maladie que sa mère.', 23, 6);
INSERT INTO EVENEMENT VALUES (18, 'Denver et Monica parte vivre a Java après la fin du bracage.', 22, 7);
INSERT INTO EVENEMENT VALUES (19, 'Denver et Monica rejoignent le professeur.', 23, 7);
INSERT INTO EVENEMENT VALUES (20, 'Helsinki et Nairobi se rejoigne dans la Pampa en Argentine.', 23, 8);
INSERT INTO EVENEMENT VALUES (21, 'le Professeur fait déverser par des dirigeables une pluie de billets.', 23, 9);
INSERT INTO EVENEMENT VALUES (22, 'le Professeur a failli empoisonner Marivi, la mère de Raquel.', 13, 10);
INSERT INTO EVENEMENT VALUES (23, 'Préparation du braquage de la fabrique de la monnaie et du timbre.', 1, 11);
INSERT INTO EVENEMENT VALUES (24, 'Découverte de la maison de Toledo par la police.', 13, 11);
INSERT INTO EVENEMENT VALUES (25, 'Préparation du casse de la banque d espagne.', 23, 12);
INSERT INTO EVENEMENT VALUES (26, 'Raquel retrouve le professeur.', 22, 13);
INSERT INTO EVENEMENT VALUES (27, 'Tokyo cherche le professeur.', 23, 13);
INSERT INTO EVENEMENT VALUES (28, 'Raquel mène la négociation avec le Professeur.', 1, 14);
INSERT INTO EVENEMENT VALUES (29, 'Le Colonel Tamayo et l inspectrice Alicia Sierra mènent les opérations pour faire arrêter le braquage.', 23, 15);
INSERT INTO EVENEMENT VALUES (30, 'La plaque du professeur pour les deux première partie.', 1, 16);
INSERT INTO EVENEMENT VALUES (31, 'La première nuit ensemble entre le Professeur et Raquel.', 12, 16);
INSERT INTO EVENEMENT VALUES (32, 'Le quartier général devient celui du Professeur et de Raquel', 23, 17);
INSERT INTO EVENEMENT VALUES (33, 'Le Professeur change de repère après que Lisbonne se soit faite capturé.', 31, 18);
```

--Insertion des données dans la table cause\_par--

```
INSERT INTO cause_par VALUES (6, 1);
INSERT INTO cause_par VALUES (5, 2);
INSERT INTO cause_par VALUES (21, 2);
INSERT INTO cause_par VALUES (2, 3);
INSERT INTO cause_par VALUES (1, 4);
INSERT INTO cause_par VALUES (2, 4);
INSERT INTO cause_par VALUES (3, 4);
INSERT INTO cause_par VALUES (5, 4);
INSERT INTO cause_par VALUES (6, 4);
INSERT INTO cause_par VALUES (7, 4);
INSERT INTO cause_par VALUES (9, 4);
INSERT INTO cause_par VALUES (10, 4);
INSERT INTO cause_par VALUES (12, 4);
INSERT INTO cause_par VALUES (13, 4);
INSERT INTO cause_par VALUES (14, 4);
INSERT INTO cause_par VALUES (15, 4);
INSERT INTO cause_par VALUES (16, 4);
INSERT INTO cause_par VALUES (1, 5);
INSERT INTO cause_par VALUES (2, 5);
INSERT INTO cause_par VALUES (3, 5);
INSERT INTO cause_par VALUES (6, 5);
INSERT INTO cause_par VALUES (7, 5);
INSERT INTO cause_par VALUES (9, 5);
INSERT INTO cause_par VALUES (10, 5);
INSERT INTO cause_par VALUES (12, 5);
INSERT INTO cause_par VALUES (13, 5);
INSERT INTO cause_par VALUES (14, 5);
INSERT INTO cause_par VALUES (15, 5);
INSERT INTO cause_par VALUES (16, 5);
INSERT INTO cause_par VALUES (1, 6);
INSERT INTO cause_par VALUES (3, 6);
INSERT INTO cause_par VALUES (1, 7);
INSERT INTO cause_par VALUES (3, 7);
INSERT INTO cause_par VALUES (3, 8);
INSERT INTO cause_par VALUES (1, 9);
INSERT INTO cause_par VALUES (1, 10);
```

```
INSERT INTO cause_par VALUES (1, 11);
INSERT INTO cause_par VALUES (2, 11);
INSERT INTO cause_par VALUES (3, 11);
INSERT INTO cause_par VALUES (4, 11);
INSERT INTO cause_par VALUES (5, 11);
INSERT INTO cause_par VALUES (6, 11);
INSERT INTO cause_par VALUES (7, 11);
INSERT INTO cause_par VALUES (8, 11);
INSERT INTO cause_par VALUES (9, 11);
INSERT INTO cause_par VALUES (10, 11);
INSERT INTO cause_par VALUES (11, 11);
INSERT INTO cause_par VALUES (12, 11);
INSERT INTO cause_par VALUES (18, 11);
INSERT INTO cause_par VALUES (17, 11);
INSERT INTO cause_par VALUES (4, 12);
INSERT INTO cause_par VALUES (8, 13);
INSERT INTO cause_par VALUES (11, 14);
INSERT INTO cause_par VALUES (1, 15);
INSERT INTO cause_par VALUES (2, 15);
INSERT INTO cause_par VALUES (3, 15);
INSERT INTO cause_par VALUES (5, 15);
INSERT INTO cause_par VALUES (6, 15);
INSERT INTO cause_par VALUES (7, 15);
INSERT INTO cause_par VALUES (9, 15);
INSERT INTO cause_par VALUES (10, 15);
INSERT INTO cause_par VALUES (12, 15);
INSERT INTO cause_par VALUES (18, 15);
INSERT INTO cause_par VALUES (17, 15);
INSERT INTO cause_par VALUES (1, 16);
INSERT INTO cause_par VALUES (4, 16);
INSERT INTO cause_par VALUES (1, 17);
INSERT INTO cause_par VALUES (4, 17);
INSERT INTO cause_par VALUES (7, 18);
INSERT INTO cause_par VALUES (10, 18);
INSERT INTO cause_par VALUES (7, 19);
INSERT INTO cause_par VALUES (10, 19);
INSERT INTO cause_par VALUES (5, 20);
INSERT INTO cause_par VALUES (9, 20);
INSERT INTO cause_par VALUES (1, 21);
INSERT INTO cause_par VALUES (1, 22);
INSERT INTO cause_par VALUES (19, 22);
INSERT INTO cause_par VALUES (1, 23);
INSERT INTO cause_par VALUES (2, 23);
INSERT INTO cause_par VALUES (4, 23);
INSERT INTO cause_par VALUES (5, 23);
INSERT INTO cause_par VALUES (6, 23);
INSERT INTO cause_par VALUES (7, 23);
INSERT INTO cause_par VALUES (8, 23);
INSERT INTO cause_par VALUES (9, 23);
INSERT INTO cause_par VALUES (10, 23);
INSERT INTO cause_par VALUES (11, 23);
INSERT INTO cause_par VALUES (1, 24);
INSERT INTO cause_par VALUES (3, 24);
INSERT INTO cause_par VALUES (17, 24);
INSERT INTO cause_par VALUES (1, 25);
INSERT INTO cause_par VALUES (2, 25);
INSERT INTO cause_par VALUES (3, 25);
INSERT INTO cause_par VALUES (5, 25);
INSERT INTO cause_par VALUES (6, 25);
INSERT INTO cause_par VALUES (7, 25);
INSERT INTO cause_par VALUES (9, 25);
INSERT INTO cause_par VALUES (10, 25);
INSERT INTO cause_par VALUES (13, 25);
INSERT INTO cause_par VALUES (14, 25);
INSERT INTO cause_par VALUES (15, 25);
INSERT INTO cause_par VALUES (16, 25);
INSERT INTO cause_par VALUES (1, 26);
INSERT INTO cause_par VALUES (3, 26);
INSERT INTO cause_par VALUES (1, 27);
INSERT INTO cause_par VALUES (2, 27);
INSERT INTO cause_par VALUES (1, 28);
INSERT INTO cause_par VALUES (3, 28);
```

```
INSERT INTO cause_par VALUES (20, 29);
INSERT INTO cause_par VALUES (21, 29);
INSERT INTO cause_par VALUES (1, 30);
INSERT INTO cause_par VALUES (1, 31);
INSERT INTO cause_par VALUES (3, 31);
INSERT INTO cause_par VALUES (1, 32);
INSERT INTO cause_par VALUES (3, 32);
INSERT INTO cause_par VALUES (1, 33);
```

```
--Insertion des données dans la table est_en_lien--  
INSERT INTO est_en_lien VALUES (1, 1, 2, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (2, 1, 3, 'Couple');  
INSERT INTO est_en_lien VALUES (3, 1, 4, 'Frère');  
INSERT INTO est_en_lien VALUES (4, 1, 5, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (5, 1, 6, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (6, 1, 7, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (7, 1, 8, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (8, 1, 9, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (9, 1, 10, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (10, 1, 11, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (11, 1, 13, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (12, 1, 14, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (13, 1, 15, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (14, 1, 16, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (15, 1, 21, 'Ennemis');  
INSERT INTO est_en_lien VALUES (16, 2, 1, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (17, 2, 3, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (18, 2, 4, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (19, 2, 5, 'Amie');  
INSERT INTO est_en_lien VALUES (20, 2, 6, 'Couple');  
INSERT INTO est_en_lien VALUES (21, 2, 7, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (22, 2, 8, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (23, 2, 9, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (24, 2, 10, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (25, 2, 11, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (26, 2, 13, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (27, 2, 14, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (28, 2, 15, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (29, 2, 16, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (30, 2, 21, 'Ennemis');  
INSERT INTO est_en_lien VALUES (31, 3, 1, 'Couple');  
INSERT INTO est_en_lien VALUES (32, 3, 2, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (33, 3, 4, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (34, 3, 5, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (35, 3, 6, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (36, 3, 7, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (37, 3, 8, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (38, 3, 9, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (39, 3, 10, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (40, 3, 11, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (41, 3, 13, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (42, 3, 14, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (43, 3, 15, 'Coéquipier');  
INSERT INTO est_en_lien VALUES (44, 3, 16, 'Coéquipier');
```

```
INSERT INTO est_en_lien VALUES (45, 3, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (46, 4, 1, 'Frère');
INSERT INTO est_en_lien VALUES (47, 4, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (48, 4, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (49, 4, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (50, 4, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (51, 4, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (52, 4, 8, 'Coéquipier');
INSERT INTO est_en_lien VALUES (53, 4, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (54, 4, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (55, 4, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (56, 4, 13, 'Ancien Amant');
INSERT INTO est_en_lien VALUES (57, 5, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (58, 5, 2, 'Amie');
INSERT INTO est_en_lien VALUES (59, 5, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (60, 5, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (61, 5, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (62, 5, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (63, 5, 8, 'Coéquipier');
INSERT INTO est_en_lien VALUES (64, 5, 9, 'Ambigu');
INSERT INTO est_en_lien VALUES (65, 5, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (66, 5, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (67, 5, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (68, 5, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (69, 5, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (70, 5, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (71, 5, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (72, 6, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (73, 6, 2, 'Couple');
INSERT INTO est_en_lien VALUES (74, 6, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (75, 6, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (76, 6, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (77, 6, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (78, 6, 8, 'Coéquipier');
INSERT INTO est_en_lien VALUES (79, 6, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (80, 6, 10, 'Coéquipier');
```

```
INSERT INTO est_en_lien VALUES (81, 6, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (82, 6, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (83, 6, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (84, 6, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (85, 6, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (86, 6, 21, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (87, 7, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (88, 7, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (89, 7, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (90, 7, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (91, 7, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (92, 7, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (93, 7, 8, 'Fils/Père');
INSERT INTO est_en_lien VALUES (94, 7, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (95, 7, 10, 'Couple');
INSERT INTO est_en_lien VALUES (96, 7, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (97, 7, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (98, 7, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (99, 7, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (100, 7, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (101, 7, 21, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (102, 8, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (103, 8, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (104, 8, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (105, 8, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (106, 8, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (107, 8, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (108, 8, 7, 'Père/Fils');
INSERT INTO est_en_lien VALUES (109, 8, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (110, 8, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (111, 8, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (112, 9, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (113, 9, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (114, 9, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (115, 9, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (116, 9, 5, 'Ambigu');
```

```
INSERT INTO est_en_lien VALUES (116, 9, 5, 'Ambigu');
INSERT INTO est_en_lien VALUES (117, 9, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (118, 9, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (119, 9, 8, 'Coéquipier');
INSERT INTO est_en_lien VALUES (120, 9, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (121, 9, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (122, 9, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (123, 9, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (124, 9, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (125, 9, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (126, 9, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (127, 10, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (128, 10, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (129, 10, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (130, 10, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (131, 10, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (132, 10, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (133, 10, 7, 'Couple');
INSERT INTO est_en_lien VALUES (134, 10, 8, 'Coéquipier');
INSERT INTO est_en_lien VALUES (135, 10, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (136, 10, 11, 'Coéquipier');
INSERT INTO est_en_lien VALUES (137, 10, 12, 'Ex');
INSERT INTO est_en_lien VALUES (138, 10, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (139, 10, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (140, 10, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (141, 10, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (142, 10, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (143, 11, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (144, 11, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (145, 11, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (146, 11, 4, 'Coéquipier');
INSERT INTO est_en_lien VALUES (147, 11, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (148, 11, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (149, 11, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (150, 11, 8, 'Coéquipier');
INSERT INTO est_en_lien VALUES (151, 11, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (152, 11, 10, 'Coéquipier');
```

```
INSERT INTO est_en_lien VALUES (157, 13, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (158, 13, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (159, 13, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (160, 13, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (161, 13, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (162, 13, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (163, 13, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (164, 13, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (165, 13, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (166, 13, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (167, 14, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (168, 14, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (169, 14, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (170, 14, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (171, 14, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (172, 14, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (173, 14, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (174, 14, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (175, 14, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (176, 14, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (177, 14, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (178, 14, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (179, 15, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (180, 15, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (181, 15, 3, 'Coéquipier');
INSERT INTO est_en_lien VALUES (182, 15, 5, 'Coéquipier');
INSERT INTO est_en_lien VALUES (183, 15, 6, 'Coéquipier');
INSERT INTO est_en_lien VALUES (184, 15, 7, 'Coéquipier');
INSERT INTO est_en_lien VALUES (185, 15, 9, 'Coéquipier');
INSERT INTO est_en_lien VALUES (186, 15, 10, 'Coéquipier');
INSERT INTO est_en_lien VALUES (187, 15, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (188, 15, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (189, 15, 16, 'Coéquipier');
INSERT INTO est_en_lien VALUES (190, 15, 21, 'Ennemis');
INSERT INTO est_en_lien VALUES (191, 16, 1, 'Coéquipier');
INSERT INTO est_en_lien VALUES (192, 16, 2, 'Coéquipier');
INSERT INTO est_en_lien VALUES (193, 16, 3, 'Coéquipier');
```

```
INSERT INTO est_en_lien VALUES (199, 16, 13, 'Coéquipier');
INSERT INTO est_en_lien VALUES (200, 16, 14, 'Coéquipier');
INSERT INTO est_en_lien VALUES (201, 16, 15, 'Coéquipier');
INSERT INTO est_en_lien VALUES (202, 16, 21, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (203, 17, 3, 'Ami');
INSERT INTO est_en_lien VALUES (204, 3, 17, 'Ami');
INSERT INTO est_en_lien VALUES (205, 19, 3, 'Mère/Fille');
INSERT INTO est_en_lien VALUES (206, 3, 19, 'Fille/Mère');
INSERT INTO est_en_lien VALUES (207, 17, 20, 'Collègue');
INSERT INTO est_en_lien VALUES (208, 17, 21, 'Collègue');
INSERT INTO est_en_lien VALUES (209, 20, 17, 'Collègue');
INSERT INTO est_en_lien VALUES (210, 20, 21, 'Collègue');
INSERT INTO est_en_lien VALUES (211, 21, 17, 'Collègue');
INSERT INTO est_en_lien VALUES (212, 21, 20, 'Collègue');
INSERT INTO est_en_lien VALUES (213, 21, 1, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (214, 21, 2, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (215, 21, 3, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (216, 21, 5, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (217, 21, 6, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (218, 21, 7, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (219, 21, 9, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (220, 21, 10, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (221, 21, 13, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (222, 21, 14, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (223, 21, 15, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (224, 21, 16, 'ENNEMIS');
INSERT INTO est_en_lien VALUES (225, 18, 1, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (226, 18, 2, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (227, 18, 3, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (228, 18, 5, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (229, 18, 6, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (230, 18, 7, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (231, 18, 9, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (232, 18, 10, 'Otage/Braqueur');
INSERT INTO est_en_lien VALUES (233, 1, 18, 'Braqueur/Otage');
INSERT INTO est_en_lien VALUES (234, 2, 18, 'Braqueur/Otage');
INSERT INTO est_en_lien VALUES (235, 3, 18, 'Braqueur/Otage');

-- 
INSERT INTO est_en_lien VALUES (237, 6, 18, 'Braqueur/Otage');
INSERT INTO est_en_lien VALUES (238, 7, 18, 'Braqueur/Otage');
INSERT INTO est_en_lien VALUES (239, 9, 18, 'Braqueur/Otage');
INSERT INTO est_en_lien VALUES (240, 10, 18, 'Braqueur/Otage');
```

# Notre répartition du travail

Afin de réaliser ce projet nous avons commencer par réfléchir tous ensemble au modèle conceptuel de données. Pendant que Malaury PEREIRA-LOUNIS créait les tables, Cathy MARTIN, Estelle BOISSERIE, et Roméo BOURDELET ont récupéré les données de la base de données et mit sur un tableau Excel partagé. Puis Cathy MARTIN et Estelle BOISSERIE ont inséré les données dans la base de données. Ensuite, nous avons réparti le travail et suivi les évolutions de chacun grâce à un groupe WhatsApp et un projet repl.it. Voici la liste des travaux réalisés par chacun :

Roméo BOURDELET a fait :

- Les déclencheurs 1 à 6,
- Les vues 1 et 2,
- Les procédures 18 et 19.

Malaury PEREIRA-LOUNIS a effectué :

- La création de toutes les tables,
- Les questions
- La préparation des requêtes (SELECT...) des questions 17, 18 et 19.

Cathy MARTIN a codé :

- Les insertions de données,
  - La procédure 24,
  - Des éléments que nous n'avons pas gardés :
- Les requêtes (SELECT...) des questions 1 à 6 et 8 à 10,
- Et une procédure nommée personne\_lier qui existait auparavant.

Estelle BOISSERIE a conçut:

- Le schéma relationnel,
- Le trigger 7,
- Toutes les fonctions,
- Les insertions de données,
- Les vues 3 et 4,
- Les réponses aux questions,
- Les procédures 1 à 23,
- Le rapport,
- Et le rendu.