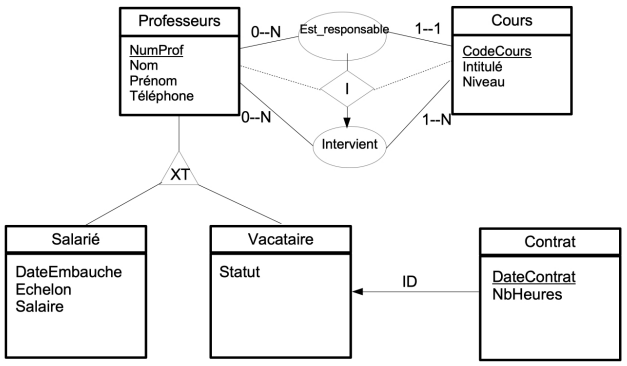
Bases de Données
Gestion des contraintes

Travaux Pratiques

Exercice 1 :

Un club de danse souhaite se doter d'une base de donnée pour gérer ses intervenants dans les cours. Après analyse, on propose le schéma Entité/Association ci-dessous.



1. Traduisez ce schéma en relationnel ; en ne tenant pas compte pour l'instant des contraintes *XT* et *I* vues plus loin. Créer les relations obtenues, ainsi que les clés et clés étrangères qui en découlent.
   Indication 1 : Puisque qu'un cours a exactement un responsable, cette association se traduira par une clé étrangère dans la relation "Cours", ne pouvant prendre la valeur NULL.
   Indication 2 : Les spécialisations et entités faibles induisent une contrainte d'existence ; par exemple, supprimer un professeur de la base induit de le supprimer également en tant que salarié ou vacataire. De même supprimer un vacataire, induit de supprimer les contrats qui s'y rapportent. Faites en sorte que toutes ces suppressions en cascade soient automatiques. (Option : ON DELETE CASCADE des clés étrangères.)
2. La contrainte *I* sur le schéma traduit le fait que si un professeur est responsable d'un cours, alors il intervient dans ce cours. C'est une implication/inclusion : tout couple (professeur, cours) de l'association "Est-Responsable" doit exister dans l'association "Intervient".
   1. Comment s'exprime cette contrainte en relationnel ? Cette contrainte peut-elle être gérée par une clé étrangère ?
   2. Créez cette contrainte dans la base de données.
   3. Créez un professeur "Jean Dupont" (sans se soucier encore qu'il soit vacataire ou salarié). Il sera responsable du cours de "SALSA" niveau débutant. Quel problème rencontrez-vous pour créer ce cours de SALSA ?
3. Une transaction [1] est un ensemble de mises à jour considérées comme une seule opération, sans limite sur le nombre d'opérations dans une même transaction. Les SGBD relationnels garantissent que les contraintes sont satisfaites AVANT et APRES la transaction. Ce qui se passe PENDANT la transaction

1. Une transaction commence par "BEGIN" et se termine par "COMMIT"

dépend des capacités des SGBD et des choix de l'opérateur. La vérification de certaines contraintes, comme les clé étrangères sous PostgreSQL, peut être différée à l'issu de la transaction.

1. Modifiez, lorsque cela est nécessaire, les déclarations de clés étrangère afin de différer leur vérification en fin de transaction.
2. Procédez maintenant à la création du cours de SALSA à l'aide d'une transaction contenant plusieurs insertions de tuples.

4. Le cahier des charges spécifie qu'un cours a obligatoirement (au moins) un intervenant (participation obligatoire à l'association "Intervient"). Peut-on considérer que cette contrainte est bien garantie dans notre base de données ?
5. La contrainte XT portant sur la double spécialisation, exprime deux choses :
   — Tous les professeurs ont la position de salarié ou de vacataire [2] (T)
   — Un vacataire ne peut pas être salarié (X)
   Créer des déclencheurs qui garantissent cette contrainte. (exemple : `https://docs.postgresql.fr/13/plpgsql-trigger.html`
6. Les contrats vacataires des années passées sont archivés, mais on considère que la relation "Intervient" ne porte que sur l'année en cours (le détail des interventions des années passées est oublié).
   Développez une fonction qui, prenant en entrée le numéro d'un professeur et le salaire horaire d'un vacataire, retourne le salaire dû sur l'ensemble de l'année pour ce professeur.

— documentation officielle de PostgreSQL : `https://docs.postgresql.fr/current/`
— Lors de la définition d'une clé étrangère, l'option "DEFERRABLE" permet de différer la vérification.
  — Pour une vérification différée, on commence la transaction concernée par "SET CONSTRAINTS nom-contrainte DEFERRED".
— Suivant les options, les déclencheurs peuvent se déclencher :
  — Pour les insertions (INSERT), les suppressions (DELETE), les mises à jour (UPDATE)
  — Avant (BEFORE) ou après (AFTER) l'évènement déclenchant
  — A chaque ligne affectée par la commande (FOR EACH ROW) ou globalement sur l'instruction (FOR EACH STATEMENT)
— Il est possible de déclarer un déclencheur comme une une contrainte (CREATE CONSTRAINT TRIGGER)
  — Dans ce cas, il peut lui même avoir la caractéristique "DEFERRABLE"
  — A condition d'être du type "FOR EACH ROW".

Le code ci-dessous construit un trigger-contrainte en insertion sur la relation "professeurs", permettant de vérifier que si un professeur est créé, il doit absolument exister dans l'une des relations "salaries" ou "vacataires".

```
drop function if exists check_fils cascade; -- supprime également en cascade les trigger qui utilise la fonction
CREATE FUNCTION check_fils() RETURNS trigger
as $$
  declare
    dans_fils  integer;
    begin
      select count(*) into dans_fils
      from ( (select  Numprof
           from salaries
          where numprof=new.numprof)
          union
          (select numprof
          from vacataires
          where numprof=new.numprof) ) t;

       if dans_fils=0 then
        raise exception 'Ce professeur doit être un vacataire ou un employé';
      end if;
        return NULL;
  end;
$$ LANGUAGE plpgsql;

create constraint trigger check_fils
after insert or update on Professeurs deferrable
for each row execute function check_fils();
```

2. Le statut d'un vacataire permet de dire s'il s'agit d'un étudiant, d'un professionnel, d'un retraité...