❑ **Exercice** : *type A*

On considère le schéma de base de données suivant, qui décrit un ensemble de fabricants de matériel informatique, les matériels qu'ils vendent, leurs clients et ce qu'achètent leurs clients. Les attributs des clés primaires des six premières relations sont soulignés.

— `Production(NomFabricant, Modele)`

— `Ordinateur(Modele, Frequence, Ram, Dd, Prix)`

— `Portable(Modele, Frequence, Ram, Dd, Ecran, Prix)`

— `Imprimante(Modele, Couleur, Type, Prix)`

— `Fabricant(Nom, Adresse, NomPatron)`

— `Client(Num, Nom, Prenom)`

— `Achat(NumClient, NomFabricant, Modele, Quantite)`

Chaque client possède un numéro unique connu de tous les fabricants. La relation `Production` donne pour chaque fabricant l'ensemble des modèles fabriquées par ce fabricant. Deux fabricants différents peuvent proposer le même matériel. La relation `Ordinateur` donne pour chaque modèle d'ordinateur la vitesse du processeur (en Hz), les tailles de la Ram et du disque dur (en GO) et le prix de l'ordinateur (en €). La relation `Portable`, en plus des attributs précédents, comporte la taille de l'écran (en pouces). La relation `Imprimante` indique pour chaque modèle d'imprimante si elle imprime en couleur (vrai/faux), le type d'impression (laser ou jet d'encre) et le prix (en €). La relation `Fabricant` stocke les noms et adresses de chaque fabricant, ainsi que le nom de son patron. La relation `Client` stocke les noms et prénoms de tous les clients de tous les fabricants. Enfin, la relation `Achat` regroupe les quadruplets (client c, fabricant f, modèle m et quantité q) tel que le client de numéro c a acheté q fois le modèle m au fabricant de nom f. On suppose que l'attribut `Quantite` est toujours strictement positif.

1. Proposer une clé primaire pour la relation `Achat`, quelles sont les conséquences en terme de modélisation ?
2. Identifier l'ensemble des clés étrangères éventuelles de chaque table.
3. Donner en SQL des requêtes répondant aux questions suivantes :
   a) Quels sont les numéros de modèles des matériels (ordinateur, portable ou imprimante) fabriqués par l'entreprise de nom Durand ?
   b) Quels sont les noms et adresses des fabricants produisant des portables dont le disque dur a une capacité d'au moins 500 Go ?
   c) Quels sont les noms des fabricants qui produisent au moins 10 modèles différents d'imprimantes ?
   d) Quels sont les numéros des clients n'ayant acheté aucune imprimante ?

❑ **Exercice** : *type B*

On dit qu'un tableau `tab` de taille n est *autoréférent* si pour tout entier `i` $\in [\![0; n-1]\!]$, `tab.(i)` est le nombre d'occurences de `i` dans `tab`. Par exemple, le tableau `ex=[| 1; 2; 1; 0 |]` est autoréférent, en effet :

— `ex.(0) = 1` et `0` apparaît bien une fois dans le tableau

— `ex.(1) = 2` et `1` apparaît bien deux fois dans le tableau

— `ex.(2) = 1` et `3` apparaît bien une fois dans le tableau

— `ex.(3) = 0` et `3` n'apparaît pas dans le tableau

1. Justifier rapidement que si `tab` est un tableau autoréférent de taille `n` alors les éléments de `tab` sont tous inférieurs ou égaux à `n`
2. Montrer que pour $n \in [\![1; 3]\!]$, il n'existe aucun tableau auto référent de taille n.
3. Déterminer un autre tableau autoréférent de taille 4 que celui donné en exemple.
4. Soit `n`≥ 7, on définit le tableau `tab` de taille `n` par :
   — `tab.(0) = n-4`
   — `tab.(1) = 2`
   — `tab.(2) = 1`
   — `tab.(n-4) = 1`

— `tab.(i) = 0` si `i` $\notin \{0, 1, 2, n-4\}$

Prouver que `tab` est autoréférent

5. Montrer que si `tab` est un tableau autoréférent de taille `n` alors la somme des éléments de `tab` vaut `n`. La réciproque est-elle vraie ?
6. Ecrire en OCaml une fonction `est_autoreferent int array -> bool` qui prend en argument un tableau d'entiers et renvoie `true` si ce tableau est autoréférent et `false` sinon. On attend une complexité en $O(n)$ où n est la taille du tableau.

   On cherche maintenant à construire un tableau autoréférent de taille `n` en utilisant un algorithme de recherche par retour sur trace (*backtracking*) qui valide une solution partielle construite jusqu'à un index `i` donné, on propose pour cela la fonction suivante :

```
(* Recherche par backtracking les tableaux autoréférents de taille n*)
let recherche n =
  let t = Array.make n (-1) in
  let rec remplir_a_partir_de i =
    if i = n then
      (* On est arrivé à la fin *)
      if est_auto t then
        affiche t
      else
        raise Echec
    else
      for k = 0 to n - 1 do
        try
          t.(i) <- k;
          (* Ici on valide que le début de solution est correcte  *)
          if not (valide t i) then raise Echec;
          (* Fin de la validation partir *)
          remplir_a_partir_de (i + 1)
        with Echec -> ()
      done
  in
  remplir_a_partir_de 0
```

7. Ecrire une fonction de validation partielle qui pour le moment renvoie toujours `true` et tester cette fonction pour de petites valeurs de $n \in \{5, 6, 7, 8\}$ (attention, il faut aussi écrire la fonction d'affichage du tableau).
8. Proposer et implémenter des améliorations de la fonction de validation partielle en utilisant les résultats établis sur les tableaux autoréférents aux questions précédentes.