Votre numéro d'anonymat : | | | |

Bases de données – 2I009

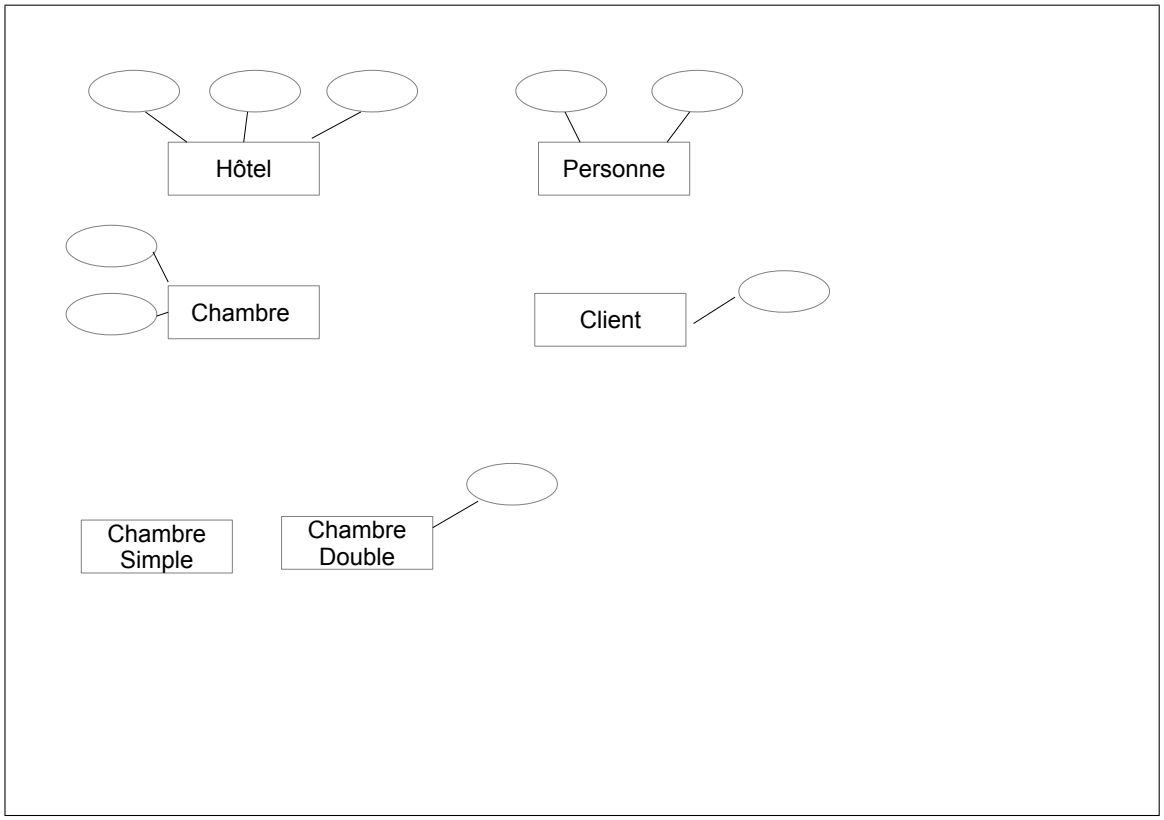
Examen du 15 Mai 2015
Durée : 2 heures – CORRIGÉ
Documents autorisés

Les téléphones mobiles doivent être éteints et rangés dans les sacs. Le barème sur 70 points (24 questions) n'a qu'une valeur indicative.

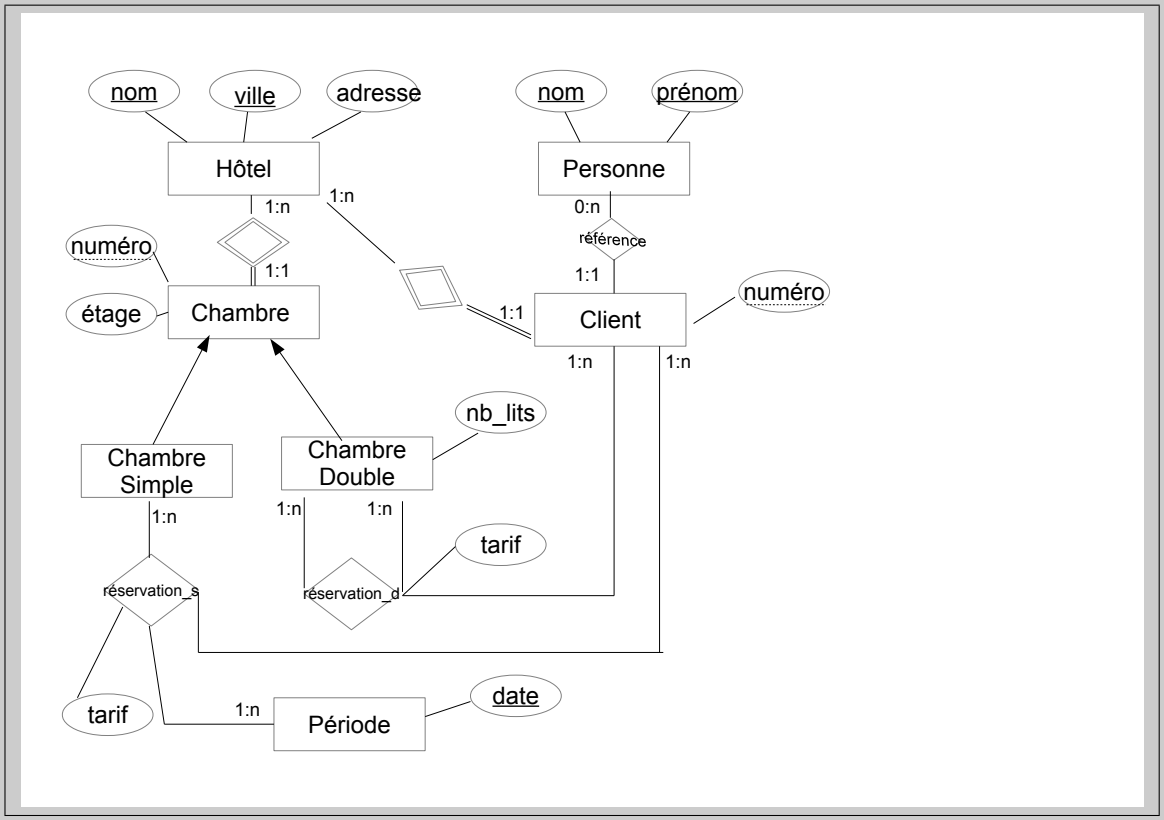
Solutions Sans Garantie

1 Schéma Entité-Association (14 pts)

On considère une base de données qui permet de gérer des hôtels avec leurs clients et qui est modélisée à l'aide d'un schéma entité-association dont une partie est illustrée dans la figure suivante :



Solution:



Attention, L'association reservation_d lie deux entités Client à une seule entité Chambre.Double

Un *Hôtel* est identifié par son nom ainsi que par le nom de la ville dans laquelle il se trouve. On connaît également son adresse.
Chaque *Personne* est identifiée par son nom et par son prénom.

Pour chaque *Client* d'un hôtel on connaît son numéro. Le numéro d'un client est unique pour un hôtel donné. Il est possible d'avoir deux clients avec le même numéro dans des hôtels différents. Un client correspond à une seule *Personne*.

Une *Chambre* a un numéro, unique dans l'hôtel auquel elle appartient, et on connaît également son étage. Une chambre peut être simple ou double. Pour une chambre double on connaît en plus le nombre de lits (1 ou 2). On connaît également les réservations en cours. La réservation d'une chambre simple est faite par un client, celle d'une chambre double par deux clients. Pour chaque réservation en cours (de chambre simple ou de chambre double) on connaît le tarif de la réservation.

Complétez le schéma Entité-Association précédent avec les informations suivantes :

Question 1 (1 point)

Précisez les attributs de chaque entité ainsi que les identifiants des entités.

Question 2 (6 points)

Ajoutez les associations suivantes en précisant leur **cardinalités** et leur éventuels **attributs** entre les entités :

A_1 : Client et Personne .
A_2 : Client et Hôtel.
A_3 : Chambre et Hôtel.
A_4 : Chambre et Chambre Simple.
A_5 : Chambre et Chambre Double.

A_6 : Client et Chambre Simple.
A_7 : Client et Chambre Double.

Question 3 (2 points)

Quelles sont les modifications à apporter à ce schéma afin qu'un client puisse réserver plusieurs fois la même chambre simple à des périodes différentes ? Modifiez le schéma précédent en conséquence.

Solution: L'éntité Période et la relation ternaire entre Client, Chambre Simple et Période.

Question 4 (5 points)

Traduisez les entités *Personne*, *Client*, *Hôtel*, *Chambre*, *Chambre Double* et les associations A_1, A_2, A_3, A_5, A_7 en relationnel. Pour chaque table, soulignez les clés primaires et ajoutez une '*' aux attributs qui sont des clés étrangères.

NB : ne **pas** donner les commandes de création de table et les types des attributs !

Réponse :

Solution:
Personne(nom, prénom)
Client(nomh*, villeh*, numéro, nomp*, prénomp*)
Hôtel(nom, ville, adresse)
Chambre(nomh*, villeh*, numéro, étage)
ChambreDouble(nomh*, villeh*, numéro, nb_lits)
Réservation($nomh_1$*, $villeh_1$*, $numero_1$*, $nomh_2$*, $villeh_2$*, $numero_2$*, nomh*, villeh*, numéro*, tarif)

2 Requêtes (42 pts)

On considère le schéma relationnel suivant :

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

La clé primaire de chaque relation est soulignée. Les attributs formant une clé étrangère sont indiqués par '*' et portent le même nom que les attributs des clés primaires référencées.

Cette base contient des informations sur des voyages proposés à des touristes dont on connait l'âge et le pays de résidence (relation *Touriste*). Chaque voyage est facturé à un prix fixe (relation *Voyage*). Chaque souscription d'un touriste à un voyage a lieu à une date donnée (relation *Souscrit*). Un voyage peut traverser plusieurs villes dans un ordre connu (relation *Traverse*), la première ville traversée ayant le numéro d'ordre **1** (ainsi, si v_i est le nom de la ville de départ du voyage de numéro v_o, le nuplet $(v_o, v_i, 1)$ appartient à *Traverse*).
On considère que chaque voyage s'arrête **exactement** *un jour par ville traversée.*

Exprimez les requêtes ci-dessous, *sans utiliser de sous-requête dans les clauses SELECT et FROM*.

Question 5 (4 points)
Quels sont les voyages coûtant moins de 1000 euros auxquels ont souscrit des touristes allemands ?

Réponse : Calcul
$\{v.\text{NumVoyage} \mid v \in$

$\}$

Quels sont les voyages coûtant moins de 1000 euros auxquels ont souscrit des touristes allemands ?

Réponse : SQL
SELECT

FROM

WHERE

Solution:

Calclul.

$$\{v.NumVoyage \mid v \in Voyage \wedge v.prix < 1000 \wedge \exists s \in Souscription(v.NumVoyage = s.NumVoyage \wedge \exists t \in Touriste(s.NumTouriste = t.NumTouriste \wedge t.paysResid =' Allemagne'))\}$$

SQL. Version Calcul.

```
select distinct v.NumVoyage
from Voyage v, Souscription s, Touriste t
where v.NumVoyage= s.NumVoyage and s.NumTouriste=t.NumTouriste
        and t.paysResid='Allemagne'
        and v.prix<1000 ;
```

Version Algèbre.

```
Select distinct NumVoyage
from Voyage Natural Join Souscription Natural Join Touriste
where paysResid='Allemagne' and prix<1000 ;
```

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

Question 6 (4 points)

Quels voyages démarrent de Paris et passent à Bordeaux puis à Strasbourg ? **NB :** un voyage peut aussi passer par d'autres villes.

Réponse : Calcul

$\{p.NumVoyage \mid p \in$

$\}$

Quels voyages démarrent de Paris et passent à Bordeaux puis à Strasbourg ?

Réponse : SQL

SELECT

FROM

WHERE

Solution: Calcul.

$\{p.NumVoyage \mid p \in Traverse \wedge p.NomVille =' Paris' \wedge \exists s, b \in Traverse(p.NumVoyage = s.NumVoyage \wedge p.NumVoyage = b.NumVoyage \wedge b.NomVille =' Bordeaux' \wedge s.NomVille =' Strasbourg' \wedge p.ordre < b.ordre \wedge b.ordre < s.ordre \wedge \neg\exists t \in Traverse(p.NumVoyage = t.NumVoyage \wedge p.ordre > t.ordre))\}$ on peut remplacer le dernier $\neg\exists$ par $\wedge p.ordre = 1$.

SQL.

```
select p.NumVoyage
from Traverse p , Traverse b, Traverse s
where p.NumVoyage=b.NumVoyage and  p.NumVoyage=s.NumVoyage
        and  p.NomVille='Paris' and b.NomVille='Bordeaux'
        and s.NomVille='Strasbourg'
        and p.ordre<b.ordre and b.ordre<s.ordre
        and not exists ( select *
                         from Traverse
                         where NumVoyage=p.NumVoyage
                                 and ordre<p.ordre );
```

on peut remplacer le not exists par : p.ordre=1

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

Question 7 (4 points)
Quels voyages ne passent (départ inclus) que par des villes de plus de $50km^2$?

Réponse : Calcul
$\{t.\text{NumVoyage} \mid t \in$

$\}$

Quels voyages ne passent (départ inclus) que par des villes de plus de $50km^2$?

Réponse : SQL
(SELECT

FROM
)

.....

(SELECT

FROM

WHERE

)

Solution: Calcul.
$\{t.NumVoyage \mid t \in Traverse \wedge \neg\exists et \in Traverse(t.NumVoyage = et.NumVoyage \wedge \exists v \in Ville(et.Ville = v.Ville \wedge v.superficie > 50))\}$

SQL Version Algèbre (première requête utilise Traverse pour avoir les voyages qui traversent au moins une ville).

```
(select NumVoyage
from Traverse)
minus
(select NumVoyage
from Traverse Natural Join Ville
where superficie <=50);
```

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

Question 8 (4 points)

Quels voyages passent par Bordeaux puis immédiatement par Toulouse ? **NB :** La ville de départ n'est pas forcément Bordeaux.

Réponse : Calcul

$\{t.\text{NumVoyage} \mid t \in$

$\}$

Quels voyages passent par Bordeaux puis immédiatement par Toulouse ?

Réponse : SQL

SELECT

FROM

WHERE

Solution: Calcul.

$$\{b.NumVoyage \mid b \in Traverse \wedge b.NomVille =' Bordeaux' \wedge \exists t \in Traverse(b.NumVoyage = t.NumVoyage \wedge t.NomVille =' Toulouse' \wedge b.ordre < t.ordre \wedge \nexists et \in Traverse(b.NumVoyage = et.NumVoyage \wedge b.ordre < et.ordre \wedge et.ordre < t.ordre))\}$$

SQL.

```
select b.NumVoyage
from Traverse b, Traverse t
where  b.NumVoyage=t.NumVoyage
       and b.NomVille='Bordeaux'
       and t.NomVille='Toulouse'
       and b.ordre<t.ordre
       and not exists ( select *
                        from Traverse
                        where NumVoyage=b.NumVoyage
                              and ordre between b.ordre
                              and t.ordre);
```

on peut remplacer le dernier not exists par : b.ordre= t.ordre-1

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

Question 9 (3 points)
Quel est l'âge moyen des touristes français ayant payé moins de 500 euros pour un voyage démarrant à Lyon et passant par Grenoble ?

Réponse : SQL
SELECT (.....) as age_moyen

FROM

WHERE

Solution:

```
select avg(fr.age) as age_moyen
from Touriste fr, Souscrit s, Voyage v, Traverse ly, Traverse gr
where fr.NumTouriste=s.NumTouriste
        and s.NumVoyage=v.NumVoyage
        and v.NumVoyage=ly.NumVoyage
        and v.NumVoyage=gr.NumVoyage
        and ly.ordre=1 and gr.ordre > 1
        and fr.paysResid='France'
        and v.prix<500;
```

Question 10 (3 points)
Quels voyages sont des circuits, c'est-à-dire démarrent et terminent dans la même ville ?

Réponse : Calcul
$\{dep.NumVoyage \mid dep \in$

}

Solution:

$$\{dep.NumVoyage \mid dep \in Traverse \wedge \exists fin \in Traverse(dep.NumVoyage = fin.NumVoyage \wedge dep.NomVille = fin.NomVille \wedge \neg\exists int \in Traverse(dep.NumVoyage = int.NumVoyage \wedge int.ordre < dep.ordre \vee int.ordre > fin.ordre))\}$$

on peut remplacer le $int.ordre < dep.ordre\vee$ dans le not exists par $\wedge dep.ordre = 1$ à l'extérieur du not exists.

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

Question 11 (4 points)
~~Quelle est la plus grande ville dans laquelle passe un touriste Portugais~~ ?

Question 12 (3 points)
~~Quels touristes ont souscrit à deux voyages à la même date ?~~

Question 13 (3 points)
~~Quelle sont les villes où démarrent le plus de voyages ?~~

Question 14 (3 points)
Quel est le rapport prix/nombre de jours pour chaque voyage ? Rappel : on considère que chaque voyage s'arrête exactement un jour par ville.

Réponse : SQL
SELECT ....., ...

FROM

.....

Solution:

```
select prix/count(*), NumVoyage as rapport_prix_duree
from Voyage Natural Join Traverse
group by NumVoyage, prix;
```

Touriste (NumTouriste, age, paysResid) **Voyage** (NumVoyage, prix)
Ville (NomVille, superficie, pays) **Souscrit** (NumTouriste*, NumVoyage*, dateSous)
Traverse (NumVoyage*, NomVille*, ordre)

Question 15 (4 points)
Quels voyages présentent le meilleur rapport prix /nombre de jours ?

Réponse : SQL
SELECT

FROM

.....

HAVING ..... = (SELECT .....(.....)

FROM

.....

.....);

Solution:

```
select NumVoyage
from Voyage Natural Join Traverse
group by NumVoyage, prix
having  prix/count(*) = (select min ( prix/count(*) )
                         from Voyage Natural Join Traverse
                         group by NumVoyage, prix);
```

Question 16 (3 points)
Quels sont les voyages souscrits uniquement par des touristes de plus de 70 ans ?

Réponse : SQL
SELECT

FROM

WHERE 70 .... (SELECT .....(.....)

FROM

WHERE);

Solution:

```
select numvoyage
from voyage v
where 70 < all (select t.age
                from touriste t, souscrit s
                where s.numtouriste=t.numtouriste  and
                      v.numvoyage=s.numvoyage);
```

3 Création de tables et contraintes (10 pts)

On considère une base décrivant les photos prises par des touristes. La base B_0 est créée par les instructions suivantes :

```
create table Lieu (
  nomLieu varchar2(20) primary key,
   latitude  number(6, 4),
  longitude  number(6, 4));

  create table Personne (
  nomPers varchar2(20) primary key,
  age number(2),
  residence  varchar2(20));

  create table Photo (
  nomPers varchar2(20) references Personne,
  num number(4),
  nomLieu varchar2(20) references Lieu,
  primary key(nomPers, num)
  );
```

```
insert into Personne(nomPers) values ('Alice');
insert into Personne(nomPers) values ('Bob');
insert into Personne(nomPers) values ('Chan');

insert into Lieu(nomLieu) values('Tour␣Eiffel');
insert into Lieu(nomLieu) values('Pont␣Neuf');
insert into Lieu(nomLieu) values('Louvres');

insert into Photo values ('Alice', 10, 'Louvres');
insert into Photo values ('Alice', 11, 'Louvres');
insert into Photo values ('Alice', 12, 'Tour␣Eiffel');
insert into Photo values ('Bob', 2, 'Tour␣Eiffel');
```

Question 17 (1 point)
Les instructions suivantes sont-elles acceptées (justifiez votre réponse) ?

Réponse :

```
insert into Personne values('Bob',22, 'Paris'),
```

Entourer : acceptée réfusée
Justification :

```
insert into Photo values ('Bob', 10, 'Louvres');
```

Entourer : acceptée refusée
Justification :

Solution: Insertion Bob : résufés car nomPersonne est une clé et Bob existe déjà
Insertion Photo de Bob : acceptée

Question 18 (1 point)
A partir de la base initiale B_0, on exécute l'instruction :

```
delete from Lieu where nomLieu='Louvres';
```

Combien de n-uplets reste-t-il dans Lieu ? Combien de n-uplets reste-t-il dans Photo ?

Réponse :
Il reste . . . . . n-uplets dans Lieu

Il reste . . . . . n-uplets dans Photo

Solution: La suppression est réfusée car par défaut la contrainte de référence est “on delete restrict”. Il reste toutes les données initiales soit 3 lieux et 4 photos. (5 photos si l’étudiant compte l’insertion précédente bien qu’on ait précisé qu’il faut partir de B_0)

Question 19 (1 point)
Que faudrait-il modifier dans la définition des tables pour ajouter la contrainte suivante : *”une personne a strictement moins de 77 ans”*. Répondez en indiquant seulement la ligne que vous modifiez.

Réponse :
Remplacer la ligne : . . . . . .

Par la ligne suivante : . . . . . .

Solution: Remplacer la ligne
Age number(2)
Par
Age number(2) **check (age < 7)**

Question 20 (1 point)
Que faudrait-il modifier dans la définition des tables pour ajouter la contrainte suivante : *”une personne réside dans un lieu existant dans la table Lieu, et le lieu de résidence est connu pour toutes les personnes”*. Répondre en indiquant seulement la ligne que vous modifiez.

Réponse :
Remplacer la ligne : . . . . . .

Par la ligne suivante : . . . . . .

Solution: Remplacer la ligne residence varchar2(20)
Par
residence varchar2(20) **not null references Lieu**

Question 21 (1 point)
Que faudrait-il modifier dans la définition des tables pour ajouter la contraintes suivante : *”pour une position géographique définie par une latitude et une longitude, il n’y a pas plus d’un lieu”*. Répondre en indiquant seulement les modification que vous faites.

Réponse :
Remplacer la ligne : . . . . . .

Par la ligne suivante : . . . . . .

Solution: Dans le bloc create table Lieu() ajouter une dernière ligne Unique(latitude, longittude)

Question 22 (2 points)
Ajouter la contrainte : *une personne a 1000 photos au maximum.*

Réponse :

Solution:

```
create assertion nbPhotos check (not exists(
select * from Photos
group by nomPers
having count(*) > 1000));
```

Question 23 (3 points)
On considère la table **Concours** pour représenter un concours de photos :

```
create table Concours (
  nomC varchar2(20),
  annee  number(4),
  lieuC varchar2(20),
  primary key(nomC, annee));
```

Définir la table **BellePhoto** indiquant la note (allant de 1 à 10) attribuée à une photo (existant dans la table **Photo**) lors d'un concours (existant dans la table **Concours**). Une personne peut présenter la même photo à plusieurs concours.

Réponse :

Solution:

```
create table BellePhoto (
  nomPers varchar2(20),
  num number(4),
  nomC varchar2(20),
  annee  number(4),
  note number(2) check(note between 1 and 10),
  foreign key (nomPers, num) references Photo,
  foreign key (nomC, annee) references Concours,
  primary key (nomPers, num, nomC, annee)
);
```

La clé n'est **pas** (nomPers, num) car une photo peut obtenir plusieurs notes à différents concours.

4 PL/SQL (4 pts)

Question 24 (4 points)

Complétez le programme ci-dessous qui affiche pour chaque lieu les personnes qui ont pris des photos et le nombre de photos qu'elles ont faites :

```
Louvres
 * Alice a fait 2 photo(s).
Pont Neuf
 * Il n'y a pas de photo.
Tour Eiffel
 * Alice a fait 1 photo(s).
 * Bob a fait 1 photo(s).
```

Réponse :

```
trouve BOOLEAN;
BEGIN
  FOR  r  IN (SELECT nomLieu FROM Lieu ORDER BY nomLieu) LOOP
```

```
    dbms_output.put_line( ____________________ );

    trouve := _____________ ;
    FOR r2 IN (SELECT nomPers, COUNT(*) AS photos FROM Photo p

               WHERE ____________________________

               GROUP BY ________________ ) LOOP

      dbms_output.put_line('* '|| _________________ ||' a fait '

                               || _________________ ||' photo(s).');

      trouve := _____________ ;
    END LOOP;
    IF(trouve = FALSE) THEN
       dbms_output.put_line('* Il n''y a pas de photo.');
    END IF;
  END LOOP;
END;
```

Solution:

```
DECLARE
trouve BOOLEAN;
BEGIN
  FOR  r  IN (SELECT nomLieu FROM Lieu ORDER BY nomLieu) LOOP
    dbms_output.put_line(r.nomLieu);
    trouve := FALSE;
    FOR r2 IN (SELECT nomPers, COUNT(*) AS photos FROM Photo p
               WHERE p.nomLieu=r.nomLieu GROUP BY nomPers ) LOOP
      dbms_output.put_line(r2.nomPers||' a fait '|| r2.photos||' photo(s).');
      trouve := TRUE;
    END LOOP;
    IF(trouve = FALSE) THEN
       dbms_output.put_line('* Il n''y a pas de photos');
    END IF;
  END LOOP;
END;
/
```