Votre numéro d'anonymat : | | | |

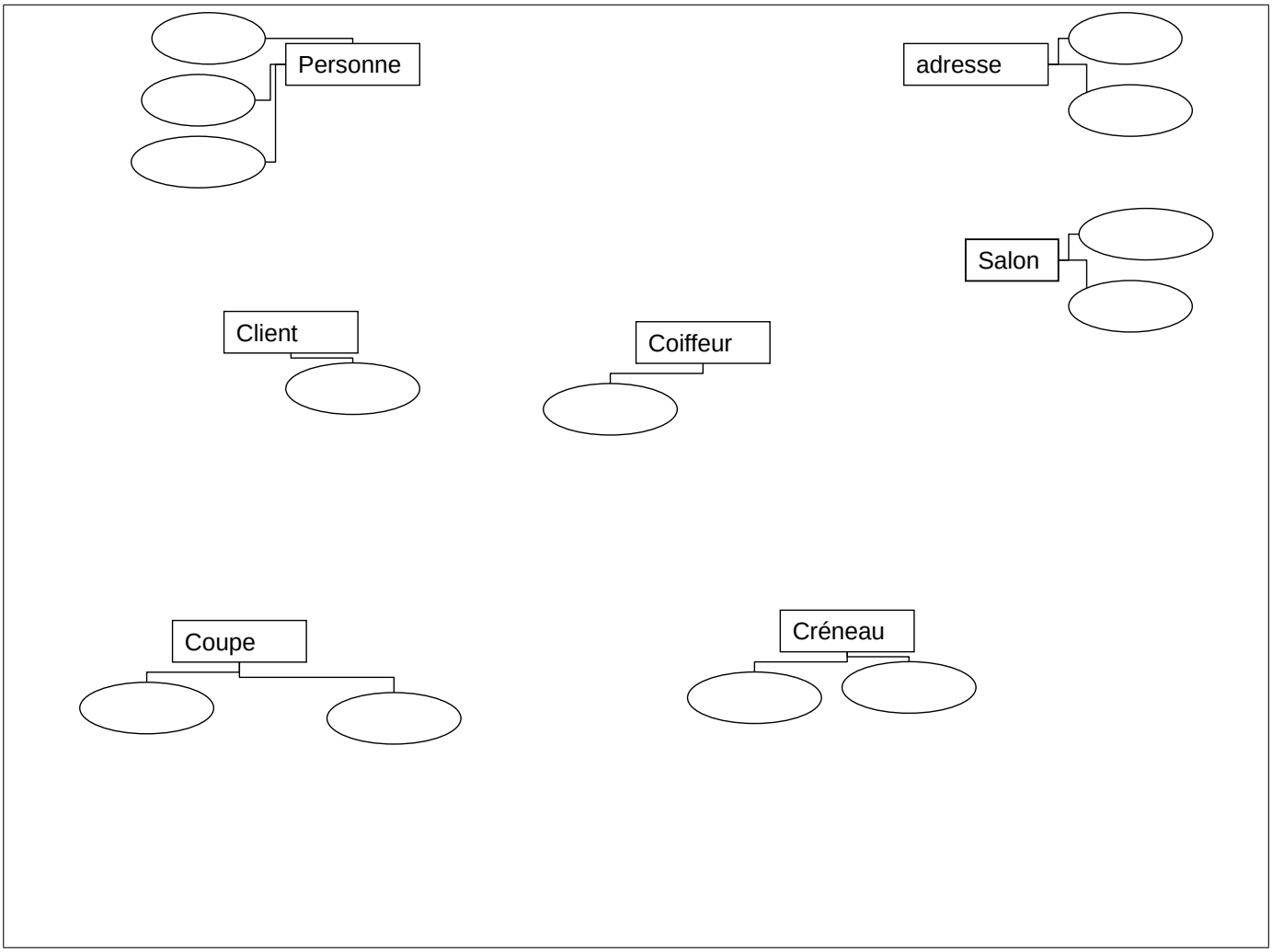
Bases de données – 2I009

Examen du 9 Mai 2016
Durée : 2 heures – CORRIGÉ
Documents autorisés

Les téléphones mobiles doivent être éteints et rangés dans les sacs. Le barème sur 60 points (16 questions) n'a qu'une valeur indicative.

1 Schéma Entité-Association (13 pts)

On considère une base de données qui permet de gérer des salons de coiffure avec leurs clients et qui est modélisée à l'aide d'un schéma entité-association dont toutes les entités apparaissent sur la figure suivante :



Une *Adresse* est composée d'une rue et d'une ville. Un *Créneau* correspnd à un jour et une heure donnés.
Chaque *Personne* est identifiée par un idpers. On connait son nom, son prénom et son adresse. Il y a deux types de personnes dans la base : Clients et Coiffeurs. Pour chaque *Client* on connait la couleur de cheveux. On connait aussi le

niveau professionnel de chaque coiffeur.

Une *Coupe* est identifiée par son nom et est d'un certain type (long, court, moyen, ...).

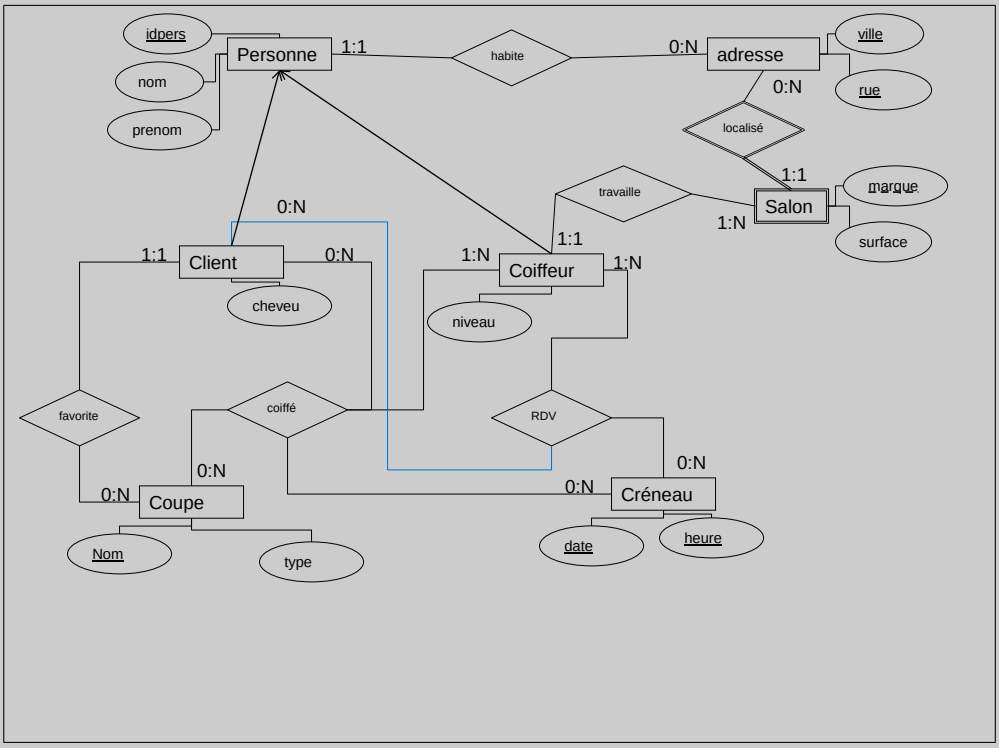
Un *Salon* de coiffure est d'une marque (Tchip, Saint Algue, JL David, ...) et a une surface donnée. Il peut y avoir plusieurs salons d'une même marque, mais un seul salon d'une marque donnée à une adresse donnée.

Question 1 (6 points)

Complétez le schéma Entité-Association précédent. Ne pas oublier de préciser les attributs de chaque entité ainsi que les identifiants des entités. Ajoutez les associations correspondant aux contraintes ci-dessous, en précisant leur **cardinalités** et leur éventuels **attributs** entre les entités. Rappel : les liens de généralisation/spécialisation (héritage) sont représentés par des flèches, les entités faibles par un double rectangle, leur association identifiante par un double losange.

— Une personne habite à une certaine adresse.
— Un salon est localisé à une certaine adresse.
— Un coiffeur travaille dans un seul salon.
— On stocke dans la base toutes les coupes réalisées (quel coiffeur a fait quelle coupe à quel client et quand)
— On stocke dans la base les rendez-vous pris par un client avec un coiffeur, à un certain créneau.
— Chaque client a une coupe favorite, celle qu'il préfère.

Solution: Barème : enlever 1/2 pt pour 2 erreurs. Les erreurs peuvent être :
— identifiant non spécifié,
— erreur dans une des cardinalités spécifiées dans l'énoncé
— erreur dans l'arité d'une association (binaire au lieu de ternaire, ...)



Question 2 (2 points)

Quelle modification doit-on apporter au schéma si un un coiffeur peut travailler dans plusieurs salons. Quel problème cela pose-t-il en ce qui concerne les rendez-vous ?

Solution: Barème : 1pt pour la réponse et 1pt pour justification

La cardinalité de Coiffeur vers Salon passe de 1 :1 à 1 :N. Problème : on ne sait pas dans quel salon un client a rendez-vous

Question 3 (5 points)

Compléter le cadre suivant pour traduire le schéma Entité/Association 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 !

Solution: Barème : enlever 1/2 pt pour 2 erreurs. Corriger par rapport à leur solution pour ne pas pénaliser deux fois les éventuelles erreurs de la question 1.

Personne(idpers, nom, prénom, rue*, ville*)
Adresse(ville, rue)
Salon(ville*, rue*, marque, surface)
Coiffeur(idpers,niveau, ville*, rue*, marque*)
Client(idpers, CoupeFavorite*)
Créneau(date,heure)
Coiffé(nomcoupe*, idclient*, idcoiffeur*, date*, heure*)
RDV(idclient*, idcoiffeur*, date*, heure*)

2 Requêtes (30 pts)

On considère le schéma relationnel suivant où les attributs de la clé primaire de chaque table sont soulignés et les attributs des clés étrangères sont indiqués par '*' et portent le même nom que les attributs des clés primaires référencées :

Serie (titre, genre, agemin, prix, distributeur, pays)
Acteur (noact, nom, prénom, naissact, pays)
Client (nocli, nom, prénom, naisscli, pays)
Episode (titre*, année, numéro, resume)
Role (titre*, année*, noact*, role)
Abo(nocli*, titre*, datedebut, nbecrans)

La table **Serie** contient des informations sur des séries télévisées avec le titre de chaque série, son genre (comédie, action, animation, ...), l'âge minimum des spectateurs, le prix d'abonnement mensuel, le distributeur et le pays. Les séries sont diffusées pendant une ou plusieurs années (saisons) et chaque saison contient des épisodes numérotés (table **Episode)** avec un résumé par épisode. La base contient également des informations sur les acteurs (**Acteur**) et les clients (**Client**) identifiés par un numéro. Pour chaque acteur et client on connaît le nom (tout en majuscule), le prénom, la date de naissance et le pays. La table **Rôle** contient pour chaque saison le rôle joué par chaque acteur. Chaque abonnement dans la table **Abo** indique le client, la série, la date de début avec le nombre d'écrans autorisés. Un abonnement est valide pour toutes les saisons d'une série à partir de l'année de début de l'abonnement. Par exemple, un abonnement qui démarre le 1er novembre 2015 pour la série "House of Cards" donne un accès à tous les épisodes de la saison 2015 et ceux des saisons suivantes.

Exprimez les requêtes ci-dessous dans les langages indiqués.
Pour SQL, ne pas utiliser les mots-clé inner join, natural join ou outer join ni d'imbrication dans le select ou le from.

Solution: regle générale :
- enlever 1/2 pt si erreur dans les prédicats
- enlever 1pt si oubli d'une négation ou d'un group by
plus de précisions sont apportées pour chaque question, si besoin.

Question 1 (4 points)

Les titres des séries distribuées par Webflux dans lesquelles l'acteur Kevin SPACEY joue un rôle pendant au moins une saison (année).

Solution: Calcul.

$$\{s.titre | Serie(s) \wedge s.distributeur =' Webflux' \wedge \exists r, a(Role(r) \wedge Acteur(a) \wedge r.titre = serie.titre \wedge a.noact = r.noact \wedge distributeur =' Webflux' \wedge a.nom =' SPACEY' \wedge a.prenom =' Kevin')\}$$

```
select titre
from Serie, Role, Acteur
where role.titre=serie.titre
  and acteur.noact=role.noact
  and distributeur='Webflux'
  and acteur.nom='SPACEY'
  and acteur.prenom='Kevin';
```

Serie (titre, genre, agemin, prix, distributeur, pays)
Acteur (noact, nom, prénom, naissact, pays)
Client (nocli, nom, prénom, naisscli, pays)
Episode (titre*, année, numéro, resume)
Role (titre*, année*, noact*, role)
Abo(nocli*, titre*, datedebut, nbecrans)

Question 2 (4 points)

Les noms des acteurs qui ont joué pendant au moins deux saisons (années) dans la série "The House of Cards".

Solution: Calcul.

$$\{a.nom \mid Acteura \wedge \exists r1, r2(Role(r1) \wedge Role(r2) \wedge a.noact = r1.noact \wedge a.noact = r2.noact \wedge r1.titre =' House\,of\,Cards' \wedge r2.titre =' House\,of\,Cards' \wedge r1.anne! = r2.anne\}$$

SQL :

```
select a.nom
from Acteur a, Role r1, Role r2
where a.noact=r1.noact and a.noact=r2.noact
  and r1.titre='House of Cards'
  and r2.titre='House of Cards'
  and r1.année<>r2.année
```

autre solution

```
select distinct a.nom
from Acteur a, Role r1
where a.noact=r1.noact
and r1.titre='House of Cards'
group by a.nom
having count(distinct a.annee) >=2 ;
```

Question 3 (4 points)

Les titres des séries allemandes sans abonnés français.

Solution:
Calcul.

$$\{s.titre \mid Serie(s) \wedge s.pays =' Allemagne' \wedge \neg\exists a, c(Abo(a) \wedge Client(c) \wedge a.titre = s.titre \wedge a.nocli = c.nocli \wedge c.pays =' France')\}$$

SQL

```
select s.titre
from Serie s
where s.pays='Allemagne'
and not exists (select * from Abo a, Client c
                where a.titre=s.titre
                  and a.nocli=c.nocli
                  and c.pays='France')
```

Serie (titre, genre, agemin, prix, distributeur, pays)
Acteur (noact, nom, prénom, naissact, pays)
Client (nocli, nom, prénom, naisscli, pays)
Episode (titre*, année, numéro, resume)
Role (titre*, année*, noact*, role)
Abo(nocli*, titre*, datedebut, nbecrans)

Question 4 (4 points)
Les titres des séries les plus anciennes (par rapport à l'année de leur premier épisode) ?

Solution: Calcul.

$$\{s.titre | Serie(s) \wedge \exists x(Episode(x) \wedge x.titre = s.titre \wedge \neg\exists e(Episode(e) \wedge x.annee > e.annee))\}$$

SQL :

```
select x.titre
from Episode x
where x.annee <= ( select annee from Episode ) ;
```

Question 5 (2 points)
On cherche des informations statistiques sur les abonnées des séries françaises. La requête retourne le titre de la série et le nombre d'abonnés par pays. Pour chaque couple (série, pays), la requête retourne, s'il y en a, le nombre d'abonnés de ce pays à cette série. Par exemple ('Les Revenants', 'Canada', 2067) indique qu'il y a 2067 abonnés canadiens pour la série "Les Revenants".

Solution:

```
select s.titre, c.pays, count(distinct a.nocli)
from Serie s, Abo a, Client c
where a.noclient=c.noclient
and s.pays='France'
and s.titre=a.titre
group by s.titre, c.pays
```

Question 6 (4 points)
Les titres des séries du genre 'Animation' avec plus de 1000 abonnés mineurs (âge < 18 ans).

Serie (titre, genre, agemin, prix, distributeur, pays)
Acteur (noact, nom, prénom, naissact, pays)
Client (nocli, nom, prénom, naisscli, pays)
Episode (titre*, année, numéro, resume)
Role (titre*, année*, noact*, role)
Abo(nocli*, titre*, datedebut, nbecrans)

Solution:

```
select s.titre
from Serie s, Client c, Abo a
where s.genre = 'Animation'
and a.titre=s.titre
and c.nocli=a.nocli
and (sysdate-c.naisscli)/365 < 18
group by s.titre
having count(distinct a.nocli) >= 1000;
```

Question 7 (4 points)
Pour chaque série, le pays avec le plus de clients abonnés à cette série.

Solution:

```
select a.titre , c.pays
from Client c, Abo a
where a.nocli=c.nocli
group by a.titre , c.pays
having count(distinct a.nocli) >=
                (select count(distinct a2.nocli)
                   from Client c2, Abo a2
                  where a2.nocli=c2.nocli
                  and a2.titre=a.titre
                   group by a2.pays)
```

Serie (titre, genre, agemin, prix, distributeur, pays)
Acteur (noact, nom, prénom, naissact, pays)
Client (nocli, nom, prénom, naisscli, pays)
Episode (titre*, année, numéro, resume)
Role (titre*, année*, noact*, role)
Abo(nocli*, titre*, datedebut, nbecrans)

Question 8 (4 points)
Les noms et les prénoms des clients qui sont abonnés à toutes les séries.

Solution: barème : si l'idée de la division n'apparait pas, alors donner 1pt seulement.

```
select c.nom, c.prenom
from Client c
where not exists (select s.titre
                  from Serie s
                 where not exists (select *
                                   from Abo a
                                  where a.nocli=c.nocli
                                    and a.tite=s.titre)
```

autre version possible en utilisant une agrégation (et d'ailleurs souvent privilégiée par les étudiants)

```
select c.nom, c.prenom
from Client c, Abonnement a
where c.nocli=a.nocli
group by c.nom, c.prenom, c.nocli
having count(a.titre) = (select count(*) from serie);
```

3 Création de tables, contraintes et mises à jour (11 pts)

On considère le schéma relationnel suivant où la clé primaire de chaque relation est soulignée et les attributs qui forment une clé étrangère sont suivis de *.

Serie (titre, genre, prixAbont) **Saison** (titre*, année) **Abonnement** (noCli*, titre*, pays)

Question 1 (5 points)

Compléter les instructions de création des tables **Serie** et **Saison** en considérant que la base accepte uniquement trois genres de séries qui sont "comédie", "action" et "animation" et qu'elle ne stocke que les saisons datant de 1990 ou plus.

Solution: 1 pt pour chaque contrainte.

```
create table serie (
 titre varchar2(20), genre varachar2(10), prix Number,
 constraint pk_serie primary key(titre),
 constraint gen_poss check(genre in ('comédie','action','animation'))
        );
create table saison (
 titre varchar2(20), année Number,
 constraint pk_saison primary key(titre , annee),
 constraint fk_serie foreign key(titre) references serie,
 constraint annee_min check(année>=1990)
        );
```

Serie (titre, genre, prixAbont) **Saison** (titre*, année) **Abonnement** (noCli*, titre*, pays)

Solution: pour les questions suivantes : enlever 1 pt si problème logique. Quelques alternatives aux solutions préconisées existent.

Question 2 (2 points)

On voudrait retirer les séries qui n'ont pas beaucoup de succès, c'est-à-dire dont le nombre d'abonnements ne dépasse pas 10. Proposer une instruction qui permet de supprimer les nuplets de Serie en conséquence.

Solution:

```
delete from serie where titre in
(select titre from Abonnement
group by titre
having count(*) <10);
```

autre solution valable

```
delete from serie S where
(select count(*) from abonnement a where a.titre=S.titre)<=10;
```

Question 3 (2 points)

On voudrait baisser de 10 % le prix d'abonnement des séries qui sont suivies par des abonnés en France. Proposer une instruction qui permet de mettre à jour la table Serie en conséquence.

Solution:

```
update Serie
set prixAbont = prixAbont *.9
where titre in (select titre from Abonnement where pays = 'France');
```

autre solution valable

```
update Serie S
set prixAbont = prixAbont *.9
where exists (select * from Abonnement a
                where S.titre=a.titre and a.pays='France');
```

Serie (titre, genre, prixAbont) **Saison** (titre*, année) **Abonnement** (noCli*, titre*, pays)

Question 4 (2 points)

On considère la table **StatsPays** dans la quelle on voudrait insérer les statistiques d'abonnement par pays et par genre de série.

StatsPays (Pays, genre, NombreAbon)

Proposer une instruction qui permet d'effectuer cette insertion.
Ne pas donner l'instruction de création la table StatsPays

Solution:

```
insert into StatsPays
(select a.pays, s.genre, count(*)
from Abonnement a, serie s
where a.titre=s.titre
group by a.pays, s.genre
);
```

4 PL/SQL (6 pts)

On considère le schéma relationnel suivant et les instances valides ci-dessous.

Serie (titre, genre, prix) **Saison** (titre*, année)
Episode (titre*, année*, numéro, résumé) **Abonnement** (noCli*, titre*, pays)

Serie

titre	genre	prix
'lucy'	'comédie'	120
'benly'	'comédie'	330
'yaris'	'action'	300

Saison

titre	annee
'lucy'	2001
'lucy'	2002
'benly'	2010

Abonnement

noCli	titre	pays
665	'lucy'	'fr'
123	'yaris'	'dz'
745	'benly'	'it'

Episode

titre	annee	numero	resume
'lucy'	2001	1	'mort de ...'
'lucy'	2001	2	'naissance ...'
'lucy'	2002	1	'depart ...' ;
'lucy'	2002	2	'arrivee ...' ;

Question 1 (6 points)

Compléter le programme suivant qui retourne, pour chaque série, le nombre de saisons, le nombre d'abonnés et

le nombre moyens d'épisode par saison. Appliqué sur la base décrite ci-dessus, le programme retourne le résultat suivant :

```
 ***La serie lucy
comporte 2 saison(s)
et 1 abonne(s)
et 2 episodes par saison en moyenne
***La serie benly
comporte 1 saison(s)
et 1 abonne(s)
et aucun episode
***La serie yaris
comporte 0 saison(s)
et 1 abonne(s)
et aucun episode
```

Solution: 1 pt par vide à remplir sauf la requete select avg(count(*)).. qui vaut 2 pts.
- enlever 1/2 pt par erreur logique dans les requêtes.
- enlever 1/2 pt pt si syntaxe d'affectation fausse, cad nbSaisons := select... au lieu de select ... into nbSaisons

```
Declare
nbSaisons number:=0;
nbAbonnes number:=0;
moyEpisode number:=0;
Begin
for s in (select * from serie)  Loop
dbms_output.put_line('***La serie '||s.titre);
select count(*) into nbSaisons from Saison where titre=s.titre;
dbms_output.put_line('comporte '||nbSaisons||' saison(s)');
select count(*) into nbAbonnes from Abonnement where titre=s.titre;
dbms_output.put_line('et '||nbAbonnes||' abonne(s)');
select avg(count(*)) into moyEpisode from  Episode e where s.titre=e.titre
        group by e.titre,e.annee;
if (moyEpisode is NOT null) then
 dbms_output.put_line('et '||moyEpisode||' episodes par saison en moyenne');
else
 dbms_output.put_line('et aucun episode');
end if;
end loop;
End;
/
```