Nom :	Prénom :	*page 1*

MLBDA – 4I801- Examen réparti du 15 novembre 2017

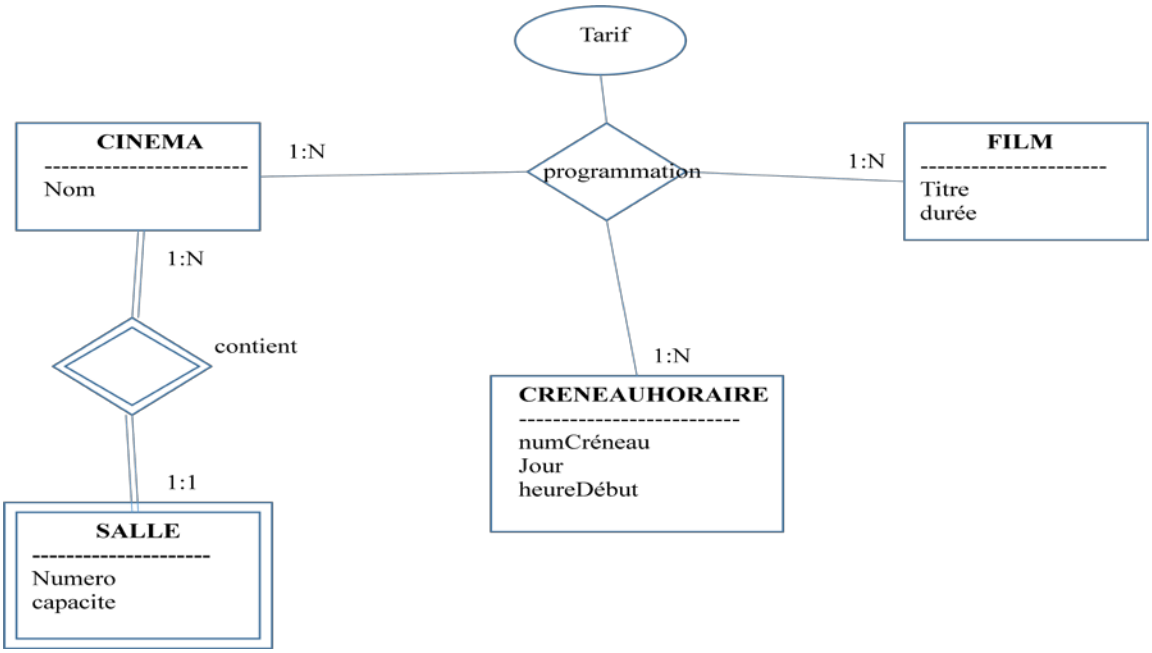
Ex1 :	Ex2 :	
Ex3 :	Ex4 :	

Seuls les documents de cours et de TD sont autorisés – Durée : 2h.

Répondre aux questions sur la feuille du sujet dans les cadres appropriés. Utiliser le dos de la feuille précédente si la réponse déborde du cadre. Le barème est donné à titre indicatif. La qualité de la rédaction sera prise en compte.

Exercice 1. Modélisation SQL3 — 4 pts

On considère le schéma Entité-Association suivant, décrivant une base de données modélisant des cinémas, des films et les séances de projection (programmation) de ces films dans les cinémas.



On veut modéliser ce schéma en SQL3. Les types suivants sont définis :

```
create type salle as object (
numSalle number,
capacite number
);
create type CreneauHoraire as object (
numCreneau number,
jour varchar2(10),
heureDebut number
);
```

Question 1. Le type Cinema est défini de la façon suivante :

```
create type Cinema as object (
nom varchar2(20),
salles enssalles,
programme pgrCine
);
```

Définissez les types `enssalles` décrivant les salles du cinéma, et `pgrCine` décrivant la programmation des films.

Question 2. Définissez le type `Film`.

Question 3. Définissez les tables **LesFilms** stockant les objets de type `Film`, et **LesCinémas** stockant les objets de type `Cinema`.

Exercice 2. DTD — 2 pts

Question 1. On veut modéliser le schéma Entité-Association de l'exercice 1 sous forme de DTD. Ecrivez la DTD décrivant ce schéma, en représentant tous les attributs du modèle Entité-Association par des attributs XML. Les tarifs des séances peuvent prendre 3 valeurs : 8€, 10€, 12€. La durée de certains films n'est pas connue. La racine du document contient uniquement les deux éléments Cinema et Film.

```
<!ELEMENT base (cinema | film)*>
<!ELEMENT cinema ……
```

Exercice 3. Xschema — 2 pts

Question 1. Donnez deux exemples de structures qu'il n'est pas possible d'exprimer avec une DTD, mais qu'on peut exprimer avec Xschema.

a)

b)

Question 2. L'affirmation suivante est-elle vraie ?

Les éléments ayant un contenu vide sont toujours de type simple (SimpleType) ?

VRAI FAUX (entourez la bonne réponse)

Exercice 4. SQL3 : insertions, requêtes et méthodes 12 pts

On considère le schéma SQL3 suivant décrivant des personnes qui suivent d'autres personnes et qui écrivent des messages. On connait la date depuis laquelle une personne suit une autre personne. Pour un message, on connait son texte, sa date de création et les tags (un ensemble de mots-clés) qui catégorisent le message. Une date s'écrit entre apostrophes '`01/01/2017`' et se compare comme un nombre avec les opérateurs habituels (<, =, >).

```
create type Personne ;
/
create type EnsTag as table of Varchar2(30);
/
create type Message as object (
  texte       varchar2(500),
  dateEcrit       Date,
  tags        EnsTag
);
/
create type EnsMessage as table of Message;
/
```

```
create type Contact as object(
  p     ref Personne,
  depuis      Date
)
/
create type EnsContact as table of Contact;
/
create type Personne as object (
  prénom      varchar2(30),
  suit        EnsContact,
  écrit       EnsMessage
);
/
```

Stockage :

Les objets *Personne* sont stockés dans la table **LesPers**.

La table LesPers contient les personnes Alice et Bob.

Question 1. Ecrivez l'instruction SQL3 permettant d'insérer dans la table LesPers la personne suivante : *Max suit Alice depuis le 01/02/2016 et a écrit le message 'Paris candidat aux JO' concernant le tag 'JO2024' le 15/09/2017.*

Question 2. Ecrivez l'instruction SQL3 qui insère Bob dans les contacts de Max le 15/11/2017.

Répondre en SQL3, en suivant le modèle du cadre réponse.

Question 3. Formuler les requêtes suivantes.

1) Quels sont les prénoms des personnes qu'Alice suit depuis le 01/01/2017 ou une date antérieure et qui ont écrit un message contenant le tag 'JO2024' ? Le résultat ne contient **pas** de doubles. La solution ne doit pas contenir de sous-requête.

```
Select _ _ _ _ _ _ _ _

From _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

     _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Where _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

2) Quelles sont les personnes qui ont écrit un message le même jour qu'Alice (i.e., Alice et la personne en réponse ont écrit au moins un message le même jour). Afficher des couples formés d'une date et d'un objet personne. Trier le résultat par date croissante.

```
Select _ _ _ _ _ _ _ _

From _ _ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _ _

     _ _ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _ _

Where _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _
```

3) Pour chaque tag, combien de personnes ont écrit au moins un message concernant ce tag ? Afficher des couples formés d'un tag et d'un nombre de personnes. Utiliser la clause group by.

```
Select _ _ _ _ _ _ _ _

From _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

     _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Group by _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _
```

4) Quels sont les couples de personnes qui se suivent mutuellement ? Afficher un ensemble de couples formés de deux prénoms. Par ex. le résultat contient (‘Alice’, Bob’) si Alice suit Bob et Bob suit Alice.

```
Select _ _ _ _ _ _ _ _

From _ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _

     _ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _

Where _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
```

5) On complète le type Personne avec la méthode *mesTags* retournant l’ensemble (sans doubles) des tags des messages d’une personne. La signature de la méthode est :
`member function mesTags return EnsTag ;`
Par exemple Bob a écrit un message ayant le tag *‘Judo’* et un autre message ayant les tags *‘Karaté’* et *‘Judo’*. Alors la méthode *mesTags()* invoquée sur Bob affiche *EnsTag(‘Karaté’, ‘Judo’)*
Ecrire le corps de la méthode.

```
member function mesTags return EnsTag is

_ _ _ _ _ _ _ _ _;

begin
_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
end;
```

6) On complète le type Personne avec la méthode *nbCommun* retournant le nombre de tags en commun entre deux personnes. La signature de la méthode est :
`Member function nbCommun(pers Personne) return Number`
Ecrire le corps de la méthode. Si possible, répondre en invoquant la méthode mesTags définie ci-dessus.

```
member function nbCommun return Number is

_ _ _ _ _ _ _ _ _

begin
_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
```

```
_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
End;
```

7) On complète le type Personne avec la méthode tagsSuivis(dist) qui retourne l'ensemble des tags concernant les personnes qu'une personne suit directement (dist=1) ou indirectment (dist >1) . Par exemple Alice suit Bob qui suit Carole. Les tags (obtenus en invoquant mesTags()) de Bob sont *('Judo' 'Karaté')* et ceux de Carole sont *('Karaté', 'Aikido').* Le résultat de *tagsSuivis(2)* invoquée sur Alice est *EnsTag('Judo' 'Karaté', 'Aikido').*

```
member function tagSuivis(dist Number) return EnsTag is

resultat   _ _ _ _ _ _ _ _ _;

begin
_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
End;
```