Devoir du 10 mars 2025 –Base de Données

Cette épreuve est composée de deux parties : partie I BD, partie II Spring/JPA. durée totale 75 mn.
Documents interdits, sauf 1 feuille A4 recto/verso

I. Partie BD (10 points)

1 Introduction

Une bibliothèque souhaite moderniser son système de gestion des emprunts et retours. Voici le schéma de la base simplifiée :

BOOKS (book_id, title, author, category, availability)
{(bid, t,a,c,av) ⇔ Un livre identifié par *bid* possède le titre *ta*, a été écrit par l'auteur *a*, appartient à la catégorie *c*. La disponibilité dans la bibliothèque est indiquée par *av*.}

MEMBERS (member_id, name, email, subscription_id, penality)
{(mid, n, e, st, p) ⇔ Un utilisateur membre de la bibliothèque est identifié par *mid*, possède un nom *n*, une adresse électronique *e*. Un utilisateur souscrit un abonnement de type *st*. Lorsqu'un livre emprunté est rendu en retard (durée d'emprunt supérieure à la durée maximale de l'abonnement *st*), une pénalité est ajoutée au total *p* des pénalités de l'emprunteur *mid*.}

SUBSCRIPTIONTYPES (subscription_id, name, max_borrow_duration, max_borrow_limit)
{(sid, n, d, l) ⇔ Un type d'abonnement à la bibliothèque est identifié par *sid*, possède le nom *n*. Un type d'abonnement est caractérisé par une durée maximale d'emprunt d (en jours) et un nombre maximal de livres empruntés simultanés *l* (c'est-à-dire le nombre de livres non encore retournés par emprunteur).}

BORROWINGS (borrow_id, member_id, book_id, borrow_date, return_date)
{(bwid,mid, bid, bd, rd) ⇔ Un emprunt de livre dans la bibliothèque est identifié par *bwid*. Le membre *mid* loue le livre *bid* de la date d'emprunt *bd* à la date de retour *rd*. La date de retour d'un emprunt en cours prend la valeur NULL. On considère qu'il y a 20% d'emprunts en cours à tout moment sur la totalité des emprunts, c'est-à-dire que 20% des emprunts ont l'attribut return_date à NULL.}

domaine(book_id, member_id, borrow_id, subscription_id) = entier >0
domaine(title, author, name, email, category) = chaine de caractères
domaine(borrow_duration, borrow_limit) = entier >0
domaine(availability) = booléen
domaine(penality) = réels ≥ 0
domaine(borrow_date, return_date) = date

$\pi_{subscription_id}$MEMBERS ⊆ $\pi_{subscription_id}$SUBSCRIPTIONTYPES
π_{member_id}BORROWINGS ⊆ π_{member_id}MEMBERS
π_{book_id}BORROWINGS ⊆ π_{book_id}BOOKS

2 Algèbre (4 points)

Vous écrirez en **algèbre** relationnelle (et pas en SQL) les requêtes 1 à 3. Lorsqu'une projection sur des attributs n'est pas explicite, cela signifie que tous les attributs d'une entité sont requis. On désire privilégier l'opération de semi-jointure lorsque cela est possible.

Q1) Obtenir la liste des membres ayant une pénalité, avec des emprunts en cours sur des livres de catégorie "SF".

Q2) Obtenir la liste des membres qui n'ont jamais emprunté de livre.

Q3) Obtenir la liste des membres ayant emprunté au moins un livre de chaque catégorie existante (les livres empruntés concernent toutes les catégories). Le schéma du résultat aura la forme (member_id).

Q4) Obtenir la liste des livres avec leurs dates d'emprunt (y compris les livres jamais empruntés). Le schéma du résultat aura la forme (book_id, borrow_date). Choisir dans les propositions celle(s) qui construise(nt) le résultat attendu.

3 SQL (4 points)

Vous écrirez en **SQL uniquement** (pas d'algèbre) les requêtes 5 et 6 :

Q5) Obtenir la liste des membres ayant emprunté au moins un livre de chaque catégorie existante (les livres empruntés concernent toutes les catégories). Le schéma du résultat aura la forme (member_id).

Q6) Obtenir la liste exacte des membres qui ont emprunté le plus de livres. Le schéma du résultat aura la forme (member_id). Une solution avec ORDER BY est refusée.

Q7) Soit l'écriture SQL de la demande " Obtenir les livres disponibles qui n'ont jamais été empruntés."

```
SELECT book_id, title, author, category
FROM BOOKS
WHERE availability = TRUE
AND book_id NOT IN (SELECT DISTINCT book_id FROM BORROWINGS);
```

Compléter l'écriture plus optimale ci-après construite à l'aide d'une jointure externe.

4 Transaction (2 points)

Soit deux sessions Oracle qui se déroulent en parallèle avec un enchainement de transactions. Les différentes opérations sont horodatées de 1 à 16. On suppose que le mode AUTOCOMMIT est désactivé dans chaque session. Une transaction est supposée commencée à la première opération SQL qui suit un COMMIT. La table BORROWINGS est notée BW dans la suite et supposée vide au départ.

Session 1	Session 2
set transaction isolation level read committed;	
1. insert into BW values (1,10,100,d1,NULL);	
	set transaction isolation level read committed;
	2. insert into BW values (2,20,200,d2,NULL);
3. commit;	
set transaction isolation level read committed;	
	4. select count(*) from BW; ①
5. select count(*) from BW; ②	
6. update BW set return_date=d3 where borrow_id=1;	
	7. delete from BW where return_date is NULL;
8. commit;	
set transaction isolation level read committed;	
	9. select count(*) from BW; ③
	10. insert into BW values (3,30,300,d4,NULL);
	11. commit;
	set transaction isolation level serializable;
	12. update BW set return_date=d5 where borrow_id=3;
13. delete from BW where borrow_id=3;	
14. commit;	
	15. select count(*) from BW; ④
	16. commit;

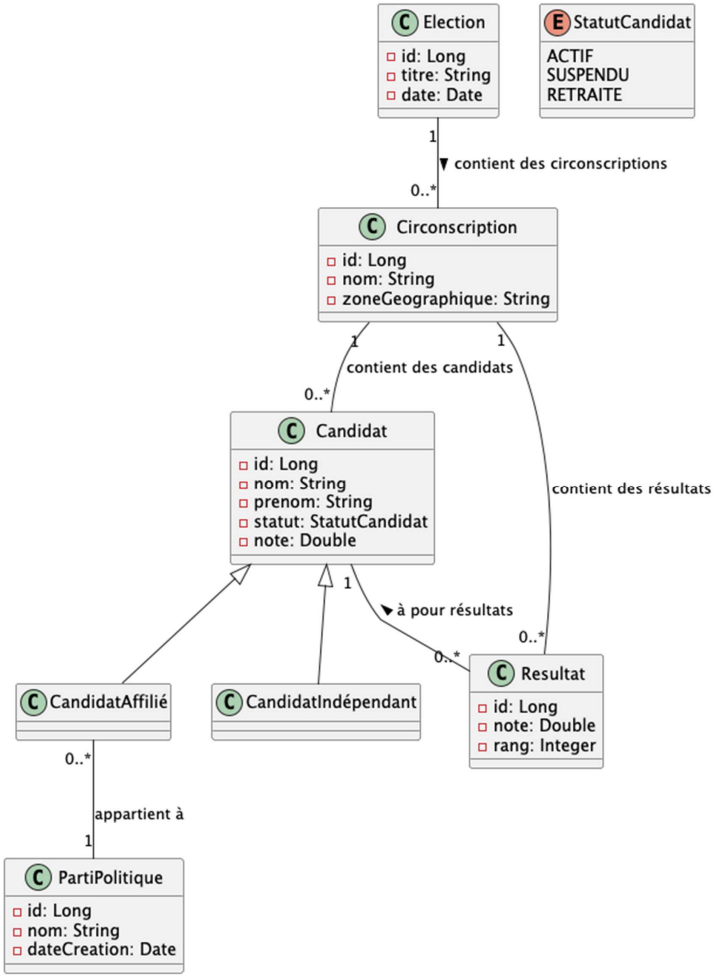
Q8) Donner le résultat renvoyé aux différents points tagués ① à ④. Le résultat correspond au résultat de la requête associée au tag.

Q9) Que se passe-t-il aux instants 7 et 13 ?

II. Mapping JPA & Repositories (10 points)

1 Contexte

Dans une France où la politique est assez instable, on nous demande de mettre à niveau le système informatique de nos institutions constitutionnelles. On nous demande de gérer les élections des candidats dans les circonscriptions. Voici la situation :



Quelques règles métier :

- La note dans les résultats doit toujours comporter deux chiffres après la virgule et ne peut pas être mise à jour une fois enregistrée dans le système.
- La note d'un candidat et d'un résultat ne doit pas être supérieure à 100. (@Max(100))
- Il doit être possible de lire les statuts des candidats en base.

Votre lead-tech a fait une petite partie à votre place, c'est-à-dire la table élection, parti politique et circonscription. Voici le code fourni :

```
@Entity
public class ElectionEntity {
    @Id
    private Long id;
    private String titre;
    private Date date;
    @OneToMany
    @JoinColumn(referencedColumnName = "id")
    private Set<CirconscriptionEntity> circonscriptionEntitySet;
}

@Entity
public class CirconscriptionEntity {
    @Id
    private Long id ;
    private String nom;
    private String ZoneGeographique;
    @OneToMany(mappedBy = "circonscriptionEntity")
    private Set<ResultatEntity> resultatEntities;
    @OneToMany(mappedBy = "circonscriptionEntity")
    private Set<CandidatEntity> candidatEntities;
}

@Entity
public class PartiePolitiqueEntity {
    @Id
    private Long id;
    private String nom;
    private Date dateCreation;
    @OneToMany(mappedBy = "partiPolitiqueEntity")
    private Set<CandidatAffilieEntity> candidatAffiliesEntities;
}
```

2 JPA

Votre lead-tech insiste sur le fait qu'il doit y avoir le moins de tables possible dans la BD. L'État n'a pas beaucoup d'agents, vous savez…

Q10) Terminer la représentation en base de données.

3 Repositories

Votre lead-tech vous laisse implémenter vos repositories ainsi que les requêtes demandées par le client..Réaliser les requêtes suivantes dans la bases de données :

- Récupérer tous les candidats indépendants.
- Obtenir les résultats d'un candidat pour une circonscription donnée.

Q11) Donner les Repositories utiles.

4 Bonus

Votre lead-tech veut vous tester !

Q12) Pourquoi mettre un type objet dans les entités pour l'ID et non un type primitif ?