Licence d'informatique – BD – l'examen du 14/06/2004 - corrigé

Durée 3h – notes manuscrites autorisées – livres interdits
Remplissez les cadres prévus (au fond gris), n'écrivez rien à l'extérieur
Ne mettez aucune information personnelle sur ce formulaire,
insérez-le dans une feuille double avec un coin cacheté

1. Des requêtes

La base de données du restaurant « Chez SQL » contient trois tables de schémas suivants (les types des attributs sont évident et omis) :

Plats (idPlat, nom, type, prix)
Vins(idVin, nom, couleur, quantité,prix)
Accord(idPlat,idVin)

11. Pour aider un client exprimez une requête SQL qui affiche tous les noms de vins rosés qui coûtent moins de 30 euros et qui s'accordent avec le gigot.

```
SELECT   v.nom
FROM     Vins v, Plats p, Accords a
WHERE    v.idVin=a.idVin AND p.idPlat=a.idPlat AND
         v.couleur= "rosé" AND v.prix<30 AND p.nom= "gigot"
```

12. Exprimez la même requête en algèbre relationnelle.

R= $\pi_{Vins.nom}(\sigma_{couleur = rosé,\ prix<30}(Vins) \triangleright\triangleleft Accords \triangleright\triangleleft \sigma_{nom=gigot}(Plats))$

13. Pour faire un destockage divisez par deux les prix de tous les vins qui ne s'accordent avec aucun plat

```
UPDATE   Vins
SET      prix=prix/2
WHERE    idVin NOT IN (SELECT idVin
                       FROM Accords)
```

14. Pour chaque plat (sauf les desserts) donnez le prix minimum du vin qui s'accorde avec ce plat

```
SELECT   p.nom, MIN (v.prix)
FROM     Vins v, Plats p, Accords a
WHERE    v.idVin=a.idVin AND p.idPlat=a.idPlat AND p.type !="dessert"
GROUP BY p.nom
```

15. Faites une liste de couples de plats qui peuvent se consommer avec le même vin rouge

```
SELECT   p1.nom, p2.nom
FROM     Accords a1, Accords a2, Plats p1, Plats p2, Vins v
WHERE    v.idVin=a1.idVin AND p1.idPlat=a1.idPlat AND
         v.idVin=a2.idVin AND p2.idPlat=a2.idPlat AND
         v.couleur= "rouge" AND p1.idPlat<p2.idPlat
```

2. Analyse de forme normale

Une bibliothèque utilise une base de données avec une table unique de schéma suivant :
Prêt(noPret, noLecteur, adresse, telephone, datePret, dateRetour, noLivre, ISBN, titre, auteur)

21. Trouvez toutes ses clés :

noPret

22. Enumérez ou représentez par un graphe les dépendances fonctionnelles

noPret→ tous les attributs ; noLecteur→adresse, telephone noLivre →ISBN→titre,Auteur

23. Cochez les cases Oui/Non et expliquez brièvement :

Est la table en :	Oui	Non	Pourquoi
1FN ?	X		Tous les attributs sont atomiques
2FN ?	X		Rien ne dépend d'une partie propre d'une clé
3FN ?		X	noLecteur→adresse est une dépendance (non-clé → non-clé)
BCFN ?		X	Comme ce n'est pas 3FN, ça ne peut être BCFN non plus

24. Donnez un exemple de redondances possibles (pensez à les encercler) dans cette BD :

noPret	noLecteur	adresse	Telephone	datePret	dateRetour	noLivre	ISBN	titre	Auteur
1	100	**Paris**	**0142030405**	14/06/04	18/06/04	173	111111	BD	Matthieu
2	100	**Paris**	**0142030405**	14/06/04	14/07/04	192	113331	SGBD	Flory
3	200	Ivry	0155555555	15/06/04	16/06/04	233	342432	BD Tintin	Hergé

25. En utilisant votre table du point précédant (24) donnez un exemple d'anomalie de modification avec quelques explications.

Supposons que le lecteur no 100 déménage à Nanterre, et son no de téléphone devient 0202020202. Si on modifie cette information seulement dans le premier tuple, ça rend la BD incohérente, parce que dans le deuxième tuple c'est toujours Paris et 0142030405 pour la même personne

26. Normalisez le schéma :

Prêt=	Prêt(<u>noPret</u>, noLecteur, datePret, dateRetour, noLivre) ▷◁ Prêt(noLecteur, adresse, telephone) ▷◁ Prêt(noLivre, ISBN) ▷◁ Prêt(ISBN, titre, auteur)

En quelle forme normale est le résultat ?	BCFN

Est-ce qu'il y a des pertes des DF ?	Non

3. JDBC – lecture de programme

31. Ajoutez les commentaires expliquant le sens et le fonctionnement du programme dans les endroits prévus

Import java.sql.*;	// on charge le package JDBC
Public class MaClasseJDBC	
{ public static void main(String args[])	
{ String url = "jdbc:odbc:entreprise";	// l'adresse de la BD
Connection con=null;	
Statement stmt;	
String query = "select NOM,PRENOM,AGE from CLIENTS";	
try	// on essaye (exceptions possibles)
{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");	// on charge le driver
Con = DriverManager.getConnection(url,"admin", "ij");	// on connecte à la BD
stmt = con.createStatement();	// on fait une zone de requête
ResultSet rs = stmt.executeQuery(query);	// on execute la requite en récupérant // le résultat dans rs
System.out.println("Liste de ???:");	
While (rs.next())	// on parcours rs
{	
String n = rs.getString(1); // nom	// on trouve le 1er, le 2ème et le 3ème //attributs du tuple courant
String p = rs.getString(2); // prenom	
int a = rs.getInt(3); // age	
if (a>40)	
{System.out.println(p + " " + n + " " + a*12);}	
}	
}	
catch(java.lang.ClassNotFoundException e)	//On traite l'exception « la classe //du driver non chargée »
{ System.err.println("Pb de driver : " + e.getMessage()); }	
catch(SQLException e)	// On traite autres exceptions de //JDBC
{ System.err.println("SQLException: " + e.getMessage()); }	
if (con!=null) try {con.close();} catch(Exception e){}	// On nettoie en fermant la connexion
}	
}	

32. Que fait ce programme ?

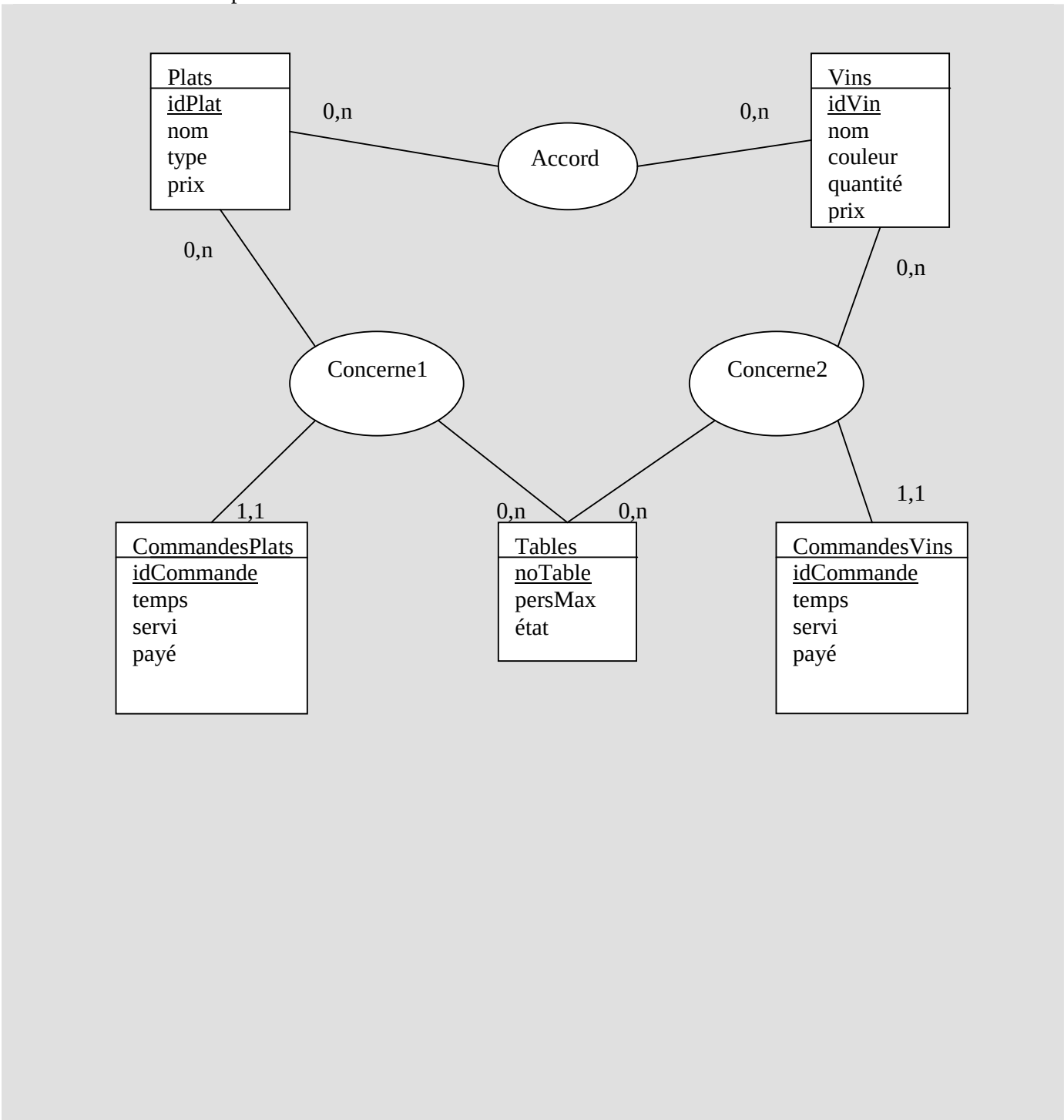
Expliquez en français	On affiche les prénoms, les noms et les ages (en mois) de tous les clients agés (>40 ans)
Expliquez en SQL en donnant une requête équivalente	SELECT prenom, nom, age*12 FROM Clients WHERE age>40

4. Conception d'une BD

On veut faire une base de données plus complète du restaurant « Chez SQL ». Cette base doit contenir les trois tables décrites dans l'exercice 1, mais également contenir l'information

- sur les tables : libre/ réservé/occupé, nombre de places
- sur les commandes de plats : quel plats, pour quelle table, pour quand, servi ou non, payé ou non
- sur les commandes de vins : même information

41. Dessinez un MCD pour cette BD



42. Déduisez le schéma de cette BD

Les trois tables de l'exo 1
et encore
Tables(noTable, persMax, etat)
CommandesPlats(noCommande, #idPlat, #noTable, temps, servi, payé)
CommandesVins(noCommande, #idVin, #noTable, temps, servi, payé)

43. Proposez trois contraintes d'intégrité non référentielle pour cette BD

1 Dans CommandesPlats et CommandesVins si c'est payé, alors c'est servi : CHECK(payé<=servi)

2 Il n'y que trois états possibles pour une table : CHECK(etat IN ('libre', 'reservé', 'occupé'))

3 Contrainte de domaine pour les attributs numériques : CHECK(prix >=0)

44. Complétez l'ordre SQL pour créer la table CommandesPlats en tenant compte de l'intégrité référentielle et non-référentielle.

```
CREATE TABLE CommandesPlats(
noCommande Integer PRIMARY KEY,
idPlat  integer FOREIGN KEY REFERENCES Plats,
noTable  integer FOREIGN KEY REFERENCES Tables,
temps Time,
 servi Integer(1),
payé Integer(1) CHECK(payé <= servi)
)
```

45. La table 7 a terminé le repas et demande l'addition. Ecrivez un programme SQL qui fait le nécessaire (imprime l'addition, note que la commande est payée et la table est libre). Soyez attentifs, ne faites pas les gens payer pour les plats déjà reglés par d'autres clients ayant utilisé la même table avant.

```
// on imprime les plats non payés avec leurs prix
SELECT p.nom, p.prix
FROM CommandesPlats c, Plats p
WHERE c.idPlat=p.idPlat AND c.noTable=7 AND c.Payé=0;

//On imprime le total
SELECT SUM(p.prix) AS 'Total de plats'
FROM CommandesPlats c, Plats p
WHERE c.idPlat=p.idPlat AND c.noTable=7 AND c.Payé=0;



// pareil pour les vins
SELECT v.nom, v.prix
FROM CommandesVins c, Vins v
WHERE c.idVin=v.idVin AND c.noTable=7 AND c.Payé=0;


//On imprime le total
SELECT SUM(v.prix) AS 'Total de vins'
FROM CommandesVins c, Vins v
WHERE c.idVin=v.idVin AND c.noTable=7 AND c.Payé=0;

// je n'imprime pas le total global…

// On note que c'est payé
UPDATE CommandesPlats
SET Payé=1
WHERE noTable=7 AND Payé=0

// On note que la table est libre

UPDATE Tables
SET etat="libre"
WHERE noTable=7
```