DISVE Licence	Année Universitaire 2008/2009 Session 2 **PARCOURS :** CSB4 & CSB6 **UE :** INF 159, Bases de données **Épreuve :** INF 159 EX **Date :** Mardi 23 juin 2009 **Heure :** 8 heure 30 **Durée :** 1 heure 30 Documents : non autorisés Épreuve de M. Alain Griffault	

Code d'anonymat :

Avertissement

- La plupart des questions sont indépendantes.
- Le barème total est de 20 points.
- Le barème de chaque question est (approximativement) proportionnel à sa difficulté.
- L'espace pour répondre est suffisant (sauf si vous l'utilisez comme brouillon, ce qui est fortement déconseillé).

Le sujet

Le sujet porte sur une gestion simplifiée des factures émises aux caisses d'un magasin.

Exercice 1 (Un premier schéma conceptuel du `Magasin` (13 points))
Un stagiaire chargé d'informatiser les factures émises aux caisses d'un magasin identifie les dépendances fonctionnelles suivantes.

DF1.1 : {`NumeroCaisse, DateHeureDebut, DateHeureFin`} → {`Caissiere, SommeDebut, SommeFin`} *qui indique que pour une caisse et pour un créneau horaire donné, la caissière et les sommes d'argent au début du créneau et à la fin du créneau sont déterminées.*

DF1.2 : {`NumeroFacture`} → {`NumeroCaisse, DateHeure, Paiement`} *qui indique que le numéro de facture détermine la caisse émettrice, sa date d'émission et le moyen de règlement.*

DF1.3 : {`NumeroFacture, Article`} → {`PrixUnitaire, Quantite`} *qui indique que pour chaque numéro de facture, l'article détermine la quantité achetée ainsi que le prix d'un article.*

Le stagiaire propose les tables suivantes, dans lesquelles il intègre quelques contraintes d'intégrité.

```
CREATE TABLE Magasin.Caisses (
  NumeroCaisse integer NOT NULL CHECK(NumeroCaisse > 0),
  Caissiere text NOT NULL,
  DateHeureDebut timestamp NOT NULL, -- timestamp = jour + heure
  DateHeureFin timestamp NOT NULL,
  SommeDebut integer NOT NULL CHECK(SommeDebut > 0),
  SommeFin integer NOT NULL,
  PRIMARY KEY (NumeroCaisse, DateHeureDebut, DateHeureFin)
);

CREATE TABLE Magasin.Factures (
  NumeroFacture integer NOT NULL CHECK(NumeroFacture > 0),
  NumeroCaisse integer NOT NULL,
  DateHeure timestamp NOT NULL,
  Paiement text NOT NULL,
  PRIMARY KEY (NumeroFacture),
  CONSTRAINT TypePaiement
    CHECK(Paiement='Espece' OR Paiement='CB' OR Paiement='Cheque')
);
```

```
CREATE TABLE Magasin.Lignesfacture (
  NumeroFacture integer NOT NULL REFERENCES Magasin.Factures(NumeroFacture),
  Article text NOT NULL,
  PrixUnitaire integer NOT NULL CHECK(PrixUnitaire > 0),
  Quantite integer NOT NULL CHECK(Quantite > 0),
  PRIMARY KEY (NumeroFacture, Article)
);
```

Question 1.1 (2 points) *Écrire la signification de toutes les contraintes décrites dans les tables.*

Le stagiaire ajoute ensuite quatre contraintes d'intégrité.

CI1.1 : La somme d'argent à la fin d'un créneau doit être supérieure à la somme en début de créneau.

CI1.2 : Un créneau horaire doit avoir une certaine durée non nulle et ne doit pas dépasser deux heures.

CI1.3 : L'heure d'une facture émise à une caisse doit correspondre à un créneau pendant lequel une caissière est bien affectée à cette caisse.

CI1.4 : Les créneaux horaires à une caisse ne doivent pas s'intersecter.

Question 1.2 (1 point) *Écrire en SQL les deux premières contraintes d'intégrité CI1.1 et CI1.2.*

Le stagiaire écrit la troisième contrainte d'intégrité CI1.3 correctement.

```
-- PostgreSQL n'accepte pas les contraintes d'integrite avec un SELECT
-- Il faut definir une fonction
CREATE FUNCTION Magasin.DateHeureCorrecte(integer, timestamp)
  RETURNS boolean AS $$
SELECT EXISTS (
  SELECT *
  FROM Magasin.Caisses
  WHERE $1 = NumeroCaisse
    AND $2 >= DateHeureDebut
    AND $2 <= DateHeureFin)
$$ LANGUAGE SQL;
-- puis appeler la fonction dans une contrainte d'integrite
ALTER TABLE Magasin.Factures ADD CONSTRAINT DateHeurePossible
  CHECK(Magasin.DateHeureCorrecte(NumeroCaisse, DateHeure) = TRUE);
```

Dans la suite vous supposerez que les trois contraintes d'intégrité sont correctement écrites.

Question 1.3 (2 points (difficile)) *En vous inspirant de l'écriture SQL de la contrainte CI1.3, écrire en SQL la contrainte d'intégrité CI1.4.*

Question 1.4 (1 point) *Indiquer et justifier si la décomposition précédente est en 3NF et/ou en BCNF.*

Question 1.5 (1 point) *Écrire la requête SQL qui liste les caisses auxquelles "Lise" a travaillé une seule fois.*

Question 1.6 (2 points) *Écrire la requête SQL qui liste les caissières ayant travaillé au moins une fois à chacune des caisses.*

Question 1.7 (2 points) *Donner l'expression en langage algébrique puis sa traduction en SQL de la requête qui liste les caissières ayant vendu au moins un article dont le prix unitaire est supérieur à 300. N'oublier pas de tenir compte de la troisième contrainte d'intégrité.*

Question 1.8 (2 points) *Écrire la requête SQL qui calcule pour chaque caissière la différence entre le total des sommes qu'elles ont encaissées (SommmeFin - SommeDebut) et la somme des factures qu'elles ont émises. Le résultat sera trié en ordre décroissant.*

Exercice 2 (Un second schéma conceptuel du `Magasin` (3 points))

L'année suivante, du fait de quelques problèmes rencontrés avec la base précédente, un nouveau stagiaire modifie les dépendances fonctionnelles et les contraintes d'intégrité pour obtenir la liste suivante de dépendances :

- *DF2.1 :* {`NumeroCaisse, DateHeureDebut, DateHeureFin`} → {`Caissiere, SommeDebut, SommeFin`} *qui indique que pour une caisse et pour un créneau horaire donné, la caissière et les sommes d'argent au début du créneau et à la fin du créneau sont déterminés. (DF2.1 = DF1.1)*
- *DF2.2 :* {`NumeroFacture`} → {`NumeroCaisse, DateHeure, Paiement`} *qui indique que le numéro de facture détermine la caisse émetrice, sa date d'émission et le moyen de règlement. (DF2.2 = DF1.2)*
- *DF2.3 :* {`NumeroFacture, Article`} → {`PrixUnitaire, Quantite`} *qui indique que pour chaque numéro de facture, l'article détermine la quantité achetée ainsi que le prix d'un article. (DF2.3 = DF1.3)*
- *DF2.4 :* {`NumeroCaisse, DateHeureDebut`} → {`DateHeureFin`} *qui indique que pour une caisse et pour un horaire de début, l'horaire de fin est déterminé.*
- *DF2.5 :* {`NumeroCaisse, DateHeureFin`} → {`DateHeureDebut`} *qui indique que pour une caisse et pour un horaire de fin, l'horaire de début est déterminé.*
- *DF2.6 :* {`Article`} → {`PrixUnitaire`} *qui indique qu'un article n'a qu'un seul prix unitaire.*
- *DF2.7 :* {`NumeroCaisse, DateHeure`} → {`DateHeureDebut, DateHeureFin`} *qui interprète la contrainte inter-relations CI1.3 comme une dépendance fonctionnelle.*

et la liste suivante de contraintes :

- *CI2.1 : La somme d'argent à la fin d'un créneau doit être supérieure à la somme en début de créneau. (CI2.1 = CI1.1)*
- *CI2.2 : Un créneau horaire doit avoir une certaine durée non nulle et ne doit pas dépasser deux heures. (CI2.2 = CI1.2)*
- *CI2.3 : Les créneaux horaires à une caisse ne doivent pas s'intersecter. (CI2.3 = CI1.4)*

Question 2.1 (3 points) *Pour les nouvelles dépendances fonctionnelles, donner et justifier une nouvelle décomposition qui soit 3NF ou bien BCNF.*

Exercice 3 (Questions de cours (4 points))

Question 3.1 (2 points) *Expliquer la différence entre état cohérent et état correct pour une base de données.*

Question 3.2 (2 points) *Expliquer l'algorithme de reprise après une panne utilisé pour restaurer une base de données dans un état correct.*