DISVE
Licence

Année Universitaire 2008/2009
Session 1 de Printemps

PARCOURS : CSB4 & CSB6
UE : INF 159, Bases de données
Épreuve : INF 159 EX
Date : Mercredi 13 mai 2009
Heure : 8 heures 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 22 points car le sujet est assez long.
– 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 horaires du tramway bordelais.

Fig. 1 – Un plan des lignes du tramway

Exercice 1 (Un premier schéma conceptuel du `Tramway` (6 points))
Un stagiaire chargé d'informatiser les horaires des tramways simplifie le problème :
– *Il ne renseigne que les stations terminus et les stations permettant un changement.*
– *Il fait l'hypothèse qu'il n'y aura jamais de tramways qui partiront tard le soir pour arriver à destination le lendemain matin. (il ne code ainsi que des heures)*
– *Il fait l'hypothèse que les horaires sont les mêmes tous les jours.*
– *Le temps d'arrêt dans les stations étant relativement court, il ne renseigne que les horaires de départ des stations, sauf pour le terminus, où l'horaire correspond à l'heure à laquelle le train repartirait s'il continuait.*

Il propose la table suivante.

```
CREATE TABLE Tramway.Horaires (
  Ligne char(1) NOT NULL,
  Depart text NOT NULL,
  Heure_Depart time NOT NULL,
  Bassens time DEFAULT NULL,
  Bassins_a_Flot time DEFAULT NULL,
  Begles time DEFAULT NULL,
  Buttiniere time DEFAULT NULL,
  Claveau time DEFAULT NULL,
  Floirac time DEFAULT NULL,
  Hotel_De_Ville time DEFAULT NULL,
  Les_Aubiers time DEFAULT NULL,
  Merignac time DEFAULT NULL,
  Pessac time DEFAULT NULL,
  Porte_Bourgogne time DEFAULT NULL,
  Quinconces time DEFAULT NULL,
  PRIMARY KEY (Ligne, Depart, Heure_Depart)
);
```

Il alimente ensuite la table avec des requêtes de la forme suivante.

```
INSERT INTO Tramway.Horaires
  (Ligne,Depart,Heure_Depart,Merignac,Hotel_De_Ville,Porte_Bourgogne,Buttiniere,Floirac)
  VALUES('A','Merignac','08:30','08:30','09:02','09:10','09:26','09:38');
INSERT INTO Tramway.Horaires
  (Ligne,Depart,Heure_Depart,Merignac,Hotel_De_Ville,Porte_Bourgogne,Buttiniere,Floirac)
  VALUES('A','Merignac','09:30','09:30','10:02','10:10','10:26','10:38');
```

Question 1.1 (1 point) *Écrire une requête SQL qui affiche les horaires ordonnés de la ligne C pour les trains allant de "Begles" aux "Aubiers".*

Question 1.2 (1 point) *Écrire une requête SQL qui affiche le nombre de trains pour chaque ligne passant par les "Quinconces" entre 8h30 et 9h30 strictement.*

Sachant que les trains roulent tous à la même vitesse sur les mêmes tronçons, la durée entre deux stations d'une même ligne est constante (par exemple 12 minutes entre "Buttiniere" et "Floirac").

Question 1.3 (1 point) *Ajouter une contrainte au schéma de la table Horaires afin que ses tuples vérifient cette contrainte de 12 minutes entre "Buttinière" et "Floirac".*

Question 1.4 (1,5 point) *Soit l'expression algébrique*

$$
\begin{array}{lll}
\pi[& & \texttt{H1.Merignac}, \texttt{H1.Hotel\_De\_Ville}, \texttt{H2.Porte\_Bourgogne}, \texttt{H2.Begles} \\
] & (\sigma[& \texttt{H1.Depart} = \texttt{'Merignac'} \ \& \\
 & & \texttt{H1.Porte\_Bourgogne} <= \texttt{H2.Porte\_Bourgogne} \ \& \\
 & & \texttt{H2.Porte\_Bourgogne} - \texttt{H1.Porte\_Bourgogne} <' 15min' \ \& \\
 & & \texttt{H2.Begles} > \texttt{H2.Porte\_Bourgogne} \\
 &] & (\alpha[A_i : H1.A_i](\texttt{Horaires}) \times \alpha[A_i : H2.A_i](\texttt{Horaires}))
\end{array}
$$

Écrire la requête SQL correspondante, puis expliquez ce qu'elle calcule.

Question 1.5 (1,5 point) *Écrire une requête SQL qui affiche les horaires ordonnés et permettant d'aller de "Bassins_a_Flot" à "Bassens" en attendant moins de 15 minutes à "Hotel_De_Ville" et en n'effectuant qu'un seul changement.*

Exercice 2 (Un second schéma conceptuel du `Tramway` (8 points))
L'année suivante, à partir des mêmes hypothèses, un nouveau stagiaire propose une conception très différente. Il considère le réseau comme un ensemble de tronçons sur lesquels circulent des trains.

```
CREATE TABLE Tramway.Troncons (
  Ligne Char(1) NOT NULL,
  Station1 text NOT NULL,
  Station2 text NOT NULL,
  Duree interval NOT NULL, -- interval : type compatible avec le type time
  PRIMARY KEY (Station1, Station2)
);

CREATE TABLE Tramway.Trains (
  Ligne Char(1) NOT NULL,
  Depart text NOT NULL,
  Heure_Depart time NOT NULL,
  Arrivee text NOT NULL,
  PRIMARY KEY (Ligne, Depart, Heure_Depart)
);
```

Il renseigne ensuite les tables.

```
INSERT INTO Tramway.Troncons VALUES('A','Merignac','Hotel_De_Ville','32min');
...
INSERT INTO Tramway.Troncons VALUES('B','Quinconces','Bassins_a_Flot','10min');
INSERT INTO Tramway.Troncons VALUES('B','Bassins_a_Flot','Claveau','8min');
INSERT INTO Tramway.Troncons VALUES('C','Begles','Porte_Bourgogne','14min');
INSERT INTO Tramway.Troncons VALUES('C','Porte_Bourgogne','Quinconces','4min');
INSERT INTO Tramway.Troncons VALUES('C','Quinconces','Les_Aubiers','14min');

INSERT INTO Tramway.Trains VALUES('A','Merignac','08:00','Bassens');
INSERT INTO Tramway.Trains VALUES('A','Merignac','08:30','Floirac');
...
INSERT INTO Tramway.Trains VALUES('A','Bassens','08:15','Merignac');
INSERT INTO Tramway.Trains VALUES('A','Floirac','08:45','Merignac');
...
INSERT INTO Tramway.Trains VALUES('B','Pessac','08:00','Bassins_a_Flot');
INSERT INTO Tramway.Trains VALUES('B','Pessac','08:30','Claveau');
...
INSERT INTO Tramway.Trains VALUES('C','Begles','08:00','Les_Aubiers');
...
```

Question 2.1 (1 point) *Ajouter une contrainte au schéma afin que les trains ne remontent pas le temps.*

Question 2.2 (1 point) *En fonction des hypothèses et de vos connaissances, donnez et justifiez une liste de dépendances fonctionnelles.*

Question 2.3 (2 points) *Pour votre liste de dépendances fonctionnelles, indiquer et justifier si la décomposition en deux tables "Troncons" et "Trains" est en 3NF et en BCNF.*

Question 2.4 (2 points) *Écrire une requête SQL qui crée une vue des trains de la ligne C de "Begles" aux "Aubiers".*

Question 2.5 (2 points) *Écrire une requête SQL qui affiche les horaires ordonnés permettant d'aller de "Bassins_a_Flot" aux "Aubiers" en attendant moins de 15 minutes aux "Quinconces" et en n'effectuant qu'un seul changement. Vous pourrez supposer avoir des vues intermédiaires (que vous nommerez) pour des lignes et/ou parties de lignes.*

Exercice 3 (Un troisième schéma conceptuel du `Tramway` (2 points))
L'année suivante, un nouveau stagiaire doit établir les nouveaux horaires mais il doit aussi intégrer toutes les stations du Tramway.

Question 3.1 (2 points) *Choisir un des deux schémas conceptuels précédents, puis décrire comment vous intégrez les nouvelles stations dans ce schéma. Justifier vos choix.*

Exercice 4 (PostGreSQL et la gestion des accès concurrents (6 points))
Le texte suivant est extrait du chapitre 13 "Contrôle d'accès simultané" de la documentation PostGreSQL.

> *Le standard SQL définit quatre niveaux d'isolation de transaction pour empêcher trois phénomènes de se produire lors de transactions simultanées. Ces phénomènes indésirables sont :*
>
> **lecture sale :** *Une transaction lit des données écrites par une transaction concurrente non validée.*
>
> **lecture non reproductible :** *Une transaction relit des données qu'elle a lues précédemment et voit que les données ont été modifiées par une autre transaction (validée depuis la lecture initiale).*

lecture fantôme : *Une transaction ré-exécute une requête renvoyant un ensemble de lignes satisfaisant une condition de recherche et trouve que l'ensemble des lignes satisfaisant la condition a changé du fait d'une autre transaction récemment validée.*

Les quatre niveaux d'isolation de transaction et les comportements correspondants sont décrits dans le tableau 4. Les quatre niveaux d'isolation définissent seulement quel phénomène ne doit pas survenir, ils ne définissent pas ce qui doit arriver. "Read Committed" est le niveau d'isolation par défaut.

Niveau d'isolation	*Lecture sale*	*non reproductible*	*Lecture fantôme*
Uncommited Read	*Possible*	*Possible*	*Possible*
Read Committed	*Impossible*	*Possible*	*Possible*
Repeatable Read	*Impossible*	*Impossible*	*Possible*
Serializable	*Impossible*	*Impossible*	*Impossible*

TAB. *1 – Niveaux d'isolation des transactions SQL*

Question 4.1 (2 points) *Donnez un exemple de problème possible lié au choix par défaut d'isolation "Read Committed" dans une base gérée par PostGreSQL.*

Ce texte est également extrait du chapitre 13 "Contrôle d'accès simultané" de la documentation PostGreSQL.

> *PostgreSQL fournit de nombreux modes de verrous pour contrôler les accès simultanés aux données des tables. La plupart des commandes PostgreSQL acquièrent automatiquement des verrous avec les modes appropriés pour s'assurer que les tables référencées ne sont pas supprimées ou modifiées de façon incompatible lorsque la commande s'exécute (par exemple, la commande ALTER TABLE ne peut pas être exécutée de façon sûre en même temps que d'autres opérations sur la même table, donc elle obtient un verrou exclusif sur la table pour s'assurer d'une bonne exécution).*
>
> *En plus des verrous au niveau table, il existe des verrous au niveau ligne, qui peuvent être des verrous exclusifs ou partagés. Un verrou exclusif sur une ligne spécifique est automatiquement acquis lorsque la ligne est mise à jour ou supprimée. Le verrou est détenu jusqu'à la fin de la transaction, que ce soit une validation ou une annulation, de la même façon que les verrous de niveau table. Les verrous au niveau ligne n'affectent pas les requêtes sur les données ; ils bloquent seulement les modifieurs d'une même ligne.*
>
> *L'utilisation de verrous explicites accroît le risque de verrous morts lorsque deux transactions (voire plus) détiennent chacune un verrou que l'autre convoite. Par exemple, si la transaction 1 a acquis un verrou exclusif sur la table A puis essaie d'acquérir un verrou exclusif sur la table B alors que la transaction 2 possède déjà un verrou exclusif sur la table B et souhaite maintenant un verrou exclusif sur la table A, alors aucun des deux ne peut continuer. PostgreSQL détecte automatiquement ces situations de blocage et les résout en annulant une des transactions impliquées, permettant ainsi à l'autre (aux autres) de se terminer (quelle est exactement la transaction annulée est difficile à prévoir mais vous ne devriez pas vous en préoccuper).*

Question 4.2 (2 points) *Expliquez la politique de gestion des inter-blocages mise en oeuvre dans PostGreSQL. Quel est le problème non traité par cette politique ?*

La documentation donne les conseils suivants aux développeurs de requêtes.

> *La meilleure défense contre les verrous morts est généralement de les éviter en s'assurant que toutes les applications utilisant une base de données acquièrent des verrous sur des objets multiples dans un ordre cohérent. Dans l'exemple ci-dessus, si les deux transactions avaient mis à jour les lignes dans le même ordre, aucun blocage n'aurait eu lieu.*

Question 4.3 (2 points) *Expliquez et donnez votre avis sur ce conseil.*