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

SUJET + CORRIGE

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".*

Réponse :

```
-- 2 versions pour les horaires de la ligne C Begles -> Les Aubiers
CREATE VIEW Tramway.reponse_1_1_V1 AS
SELECT DISTINCT Begles, Porte_Bourgogne, Quinconces, Les_Aubiers
FROM            Tramway.Horaires
WHERE           Ligne = 'C'
  AND           Begles < Les_Aubiers
ORDER BY        Begles;
CREATE VIEW Tramway.reponse_1_1_V2 AS
SELECT DISTINCT Begles, Porte_Bourgogne, Quinconces, Les_Aubiers
FROM            Tramway.Horaires
WHERE           Depart = 'Begles'
ORDER BY        Begles;
```

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.*

Réponse :

```
-- Le nombre de trains pour chaque ligne passant par les quinconces entre 8h30 et 9h30 strictement
CREATE VIEW Tramway.reponse_1_2 AS
SELECT DISTINCT Ligne, count(*)
FROM            Tramway.Horaires
WHERE           Quinconces > '08:30'
  AND           Quinconces < '09:30'
GROUP BY        Ligne
ORDER BY        Ligne;
```

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".*

Réponse :

```
-- Le troncon entre Buttiniere et Floirac prend 12 minutes
ALTER TABLE Tramway.Horaires ADD CONSTRAINT troncon1
  CHECK ((Depart='Merignac' AND Floirac-Buttiniere='12min') OR
         (Depart='Floirac' AND Buttiniere-Floirac='12min'));
```

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

$$\pi[\ \ H1.Merignac, H1.Hotel_De_Ville, H2.Porte_Bourgogne, H2.Begles$$
$$](\sigma[\ \ H1.Depart = \text{'Merignac'}\ \&$$
$$H1.Porte_Bourgogne <= H2.Porte_Bourgogne\ \&$$
$$H2.Porte_Bourgogne - H1.Porte_Bourgogne <' 15min'\ \&$$
$$H2.Begles > H2.Porte_Bourgogne$$
$$](\alpha[A_i : H1.A_i](Horaires) \times \alpha[A_i : H2.A_i](Horaires))$$

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

Réponse :

```
-- Les horaires de la ligne Merignac -> Begles avec 1 changement
-- et une attente de moins de 15 minutes Porte_Bourgogne
CREATE VIEW Tramway.reponse_1_4 AS
SELECT DISTINCT H1.Merignac, H1.Hotel_De_Ville, H2.Porte_Bourgogne, H2.Begles
FROM            Tramway.Horaires H1, Tramway.Horaires H2
WHERE           H1.Depart= 'Merignac'
  AND           H1.Porte_Bourgogne <= H2.Porte_Bourgogne
  AND           H2.Porte_Bourgogne - H1.Porte_Bourgogne < '15min'
  AND           H2.Porte_Bourgogne < H2.Begles;
```

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.*

Réponse :

```
-- Les horaires des trains entre Bassins_a_Flot et Bassens avec 1 changement
-- et une attente de moins de 15 minutes Hotel_De_Ville
CREATE VIEW Tramway.reponse_1_5 AS
SELECT DISTINCT H1.Bassins_a_Flot, H2.Hotel_De_Ville, H2.Porte_Bourgogne, H2.Buttiniere, H2.Bassens
FROM            Tramway.Horaires H1, Tramway.Horaires H2
WHERE           H1.Bassins_a_Flot < H1.Hotel_De_Ville
  AND           H1.Hotel_De_Ville <= H2.Hotel_De_Ville
  AND           H2.Hotel_De_Ville - H1.Hotel_De_Ville < '15min'
  AND           H2.Hotel_De_Ville < H2.Bassens
ORDER BY        H1.Bassins_a_Flot;
```

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.*

Réponse :

```
-- Les trains ne remontent pas le temps
ALTER TABLE Tramway.Troncons ADD CONSTRAINT temps
  CHECK (Duree > '0min');
```

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

Réponse : *D'après la clef primaire de la table du premier stagiaire :*

DF11 : $\{Ligne,\ Depart,\ Heure_Depart\} \rightarrow \{Toutes\ les\ stations\}$

D'après les clefs primaires des tables du second stagiaire :

DF21 : $\{Station1,\ Station2\} \rightarrow \{Duree\}$

DF22 : $\{Station1,\ Station2\} \rightarrow \{Ligne\}$

DF23 : $\{Ligne,\ Depart,\ Heure_Depart\} \rightarrow \{Arrivee\}$

D'après mes connaissances, à Bordeaux, un tronçon identifie une ligne, mais le partage d'un tronçon par plusieurs lignes est tout à fait possible en pratique. Le fait que $\{Station1,\ Station2\} \rightarrow \{Ligne\}$ *interdit cette possibilité. A Bordeaux, une ligne n'indique pas le sens de parcours, et plusieurs variantes correspondent à la même ligne. Il aurait été possible de différencier les noms des lignes en fonction des gares de départ et d'arrivée : cela ajouterai les dépendances fonctionnelles.*

DFH1 : $\{Ligne\} \rightarrow \{Depart,\ Arrivee\}$

DFH2 : $\{Depart,\ Arrivee\} \rightarrow \{Ligne\}$

D'autres DFs étaient possibles.

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.*

Réponse : *En considérant les dépendances DF21, DF22 et DF23, les tables "Troncons" et "Trains" sont BCNF. Si l'on ajoute les dépendances DFH1 et DFH2, la table "Trains" n'est pas 2NF, car un attribut de la clef est un déterminant. Il faut donc décomposer la tables "trains" en deux tables "Lignes" et "HorairesDepart".*

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

"Lignes" et "HorairesDepart" sont en BCNF. La décomposition est sans perte du fait qu'elle provient des dépendances DFH1 et DFH2.

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

Réponse :

```
-- Les trains ligne C de Begles aux Aubiers
-- La question n'est pas assez precise, deux interprtions.
-- Une premiere reponse simple avec les seules stations depart et arrivee
CREATE VIEW Tramway.reponse_2_4_a AS
SELECT DISTINCT *
FROM            Tramway.Trains
WHERE           Ligne  = 'C'
  AND           Depart = 'Begles'
  AND           Arrivee = 'Les_Aubiers'
ORDER BY        Heure_Depart;
-- Une seconde reponse avec toutes les stations
CREATE VIEW Tramway.reponse_2_4 AS
SELECT DISTINCT Trains.Ligne,
  Heure_Depart AS Begles,
  Heure_Depart + Tr1.Duree AS Porte_Bourgogne,
  Heure_Depart + Tr1.Duree + Tr2.Duree AS Quinconces,
  Heure_Depart + Tr1.Duree + Tr2.Duree + Tr3.Duree AS Les_Aubiers
FROM            Tramway.Troncons Tr1,
                Tramway.Troncons Tr2,
                Tramway.Troncons Tr3,
                Tramway.Trains
WHERE           'Begles' = Trains.Depart
  AND           Trains.Depart = Tr1.Station1
  AND           Tr1.Station2 = Tr2.Station1
  AND           Tr2.Station2 = Tr3.Station1
  AND           Tr3.Station2 = 'Les_Aubiers'
ORDER BY        Trains.Heure_Depart;
```

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.*

Réponse :

```
-- Les trains ligne B de Bassins_a_Flot aux Quinconces
CREATE VIEW Tramway.reponse_2_5a AS
SELECT DISTINCT Trains.Ligne,
                Heure_Depart AS Bassins_a_Flot,
                Heure_Depart + Troncons.Duree AS Quinconces
FROM            Tramway.Troncons,
                Tramway.Trains
```

```
WHERE           Trains.Ligne = 'B'
  AND           Troncons.Ligne = 'B'
  AND           'Bassins_a_Flot' = Trains.Depart
  AND           Trains.Depart = Troncons.Station2
  AND           Troncons.Station1 = 'Quinconces'
UNION
SELECT DISTINCT Trains.Ligne,
                Heure_Depart + Tr1.Duree AS Bassins_a_Flot,
                Heure_Depart + Tr1.Duree + Tr2.Duree AS Quinconces
FROM            Tramway.Troncons AS Tr1,
                Tramway.Troncons AS Tr2,
                Tramway.Trains
WHERE           Trains.Ligne = 'B'
  AND           Tr1.Ligne = 'B'
  AND           Tr2.Ligne = 'B'
  AND           'Claveau' = Trains.Depart
  AND           Trains.Depart = Tr1.Station2
  AND           Tr1.Station1 = Tr2.Station2
  AND           Tr2.Station1 = 'Quinconces';

-- Les trains ligne C de Quinconces aux Aubiers
CREATE VIEW Tramway.reponse_2_5b AS
SELECT DISTINCT Trains.Ligne,
             Heure_Depart AS Quinconces,
             Heure_Depart + Tr1.Duree + Tr2.Duree + Tr3.Duree AS Les_Aubiers
FROM            Tramway.Troncons AS Tr1,
                Tramway.Troncons AS Tr2,
                Tramway.Troncons AS Tr3,
                Tramway.Trains
WHERE           Trains.Ligne = 'C'
  AND           Tr1.Ligne = 'C'
  AND           Tr2.Ligne = 'C'
  AND           Tr3.Ligne = 'C'
  AND           'Begles' = Trains.Depart
  AND           Trains.Depart = Tr1.Station1
  AND           Tr1.Station2 = Tr2.Station1
  AND           Tr2.Station2 = Tr3.Station1
  AND           Tr3.Station2 = 'Les_Aubiers';

-- Les horaires des trains entre Bassins_a_Flot et Les_Aubiers avec 1 changement
-- et une attente de moins de 15 minutes aux Quinconces
CREATE VIEW Tramway.reponse_2_5 AS
SELECT DISTINCT TB.Bassins_a_Flot,
                TC.Quinconces,
                TC.Les_Aubiers
FROM            Tramway.reponse_2_5a AS TB,
                Tramway.reponse_2_5b AS TC
WHERE           TB.Quinconces <= TC.Quinconces
  AND           TC.Quinconces - TB.Quinconces < '15min'
ORDER BY        TB.Bassins_a_Flot;
```

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.*

Réponse : *Le premier schéma conceptuel est simple, mais il souffre de plusieurs défauts :*
- *Utilisation de la valeur "NULL" pour les stations non concernées par un train. La valeur "NULL" peut poser des problèmes dans certaines requêtes.*
- *Redondance d'informations pour les horaires.*

– Il est très difficile d'écrire une contrainte d'intégrité garantissant que les horaires ont été saisis correctement. Le second schéma (si l'on considère que la dépendance DFH1 n'existe pas) est le bon schéma pour notre problème. Son défaut est une difficulté plus grande pour définir les vues des horaires par ligne, mais ce travail n'est à faire qu'une fois.
Si le troisième stagiaire à des compétences en Bases de Données, il choisira le second schéma comme base, sinon il construira un tableau "excel" de plus.
L'intégration la plus simple consiste à dire qu'une station appartient à un tronçon, et qu'elle est située à une certaine distance d'une extrémité.

```
CREATE TABLE Tramway.Stations (
  Nom text NOT NULL,        -- Nom de la station
  Station1 text NOT NULL,   -- une station d'un troncon
  Station2 text NOT NULL,   -- l'autre station du troncon
  Duree interval NOT NULL,  -- duree entre Nom et Station1
  PRIMARY KEY (Nom)
);
```

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.*

Réponse : *Voir chapitre 11 du poly, sous-sections 11.1.1 et 11.1.2.*

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 ?*

Réponse : *C'est la politique "Détection et correction". Voir chapitre 11 du poly, sous-section 11.5.1.*

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.*

Réponse : *Cette solution est correcte, mais inapplicable en pratique.*
Elle est correcte car si effectivement les tuples sont "ordonnés" et que les verrous sont acquis en ordre croissant des tuples, il est impossible de créer un cycle dans le graphe d'attente.
Elles est inapplicable, car elle souffre de deux défauts majeurs :
- *une rigueur d'organisation pour que tous ceux chargés d'écrire des requêtes sur la base soient au courant de cet ordre.*
- *une non optimisation des verrous qui vont devoir être conservés beaucoup plus longtemps que nécessaire.*