DISVE Licence	Année Universitaire 2010/2011 Session 1 de Printemps **PARCOURS :** CSB4 & CSB6 **UE :** INF 159, Bases de données **Épreuve :** INF 159 EX **Date :** Mardi 3 mai 2010 **Heure :** 11 heures **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 23 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).

Exercice 1 (Conception et SQL (12 points))

L'exercice porte sur une gestion simplifiée d'arbres généalogiques. Le concepteur de la base a conçu un schéma relationnel composé des trois relations `Personnes, Parents` *et* `Mariages`*. La première partie de l'exercice consiste à comprendre et à expliquer ce schéma. La seconde partie sera composée de requêtes, et la troisième de variantes. Les sources sont au format* `PostGreSQL`*, et toutes les relations sont préfixées par le nom du schéma* `Genealogie`*.*

```
CREATE SCHEMA Genealogie;

CREATE TABLE Genealogie.Personnes (
  -- Typage des attributs
  Id serial NOT NULL,          -- serial = sequence d'entier 1, 2, 3, ...
  Nom text NOT NULL,
  Prenom text NOT NULL,
  Sexe char NOT NULL CHECK (Sexe = 'F' or Sexe = 'M'),
  DateNaissance date NOT NULL,
  DateDeces date NOT NULL DEFAULT 'infinity',
  -- Clefs candidates
  PRIMARY KEY (id),
  -- Contraintes d'integrite elementaire
  CHECK (DateNaissance < DateDeces)
);
```

Question 1.1 (1 point) *Pour la relation* `Personnes`*, donnez la seule dépendance fonctionnelle déclarée, puis expliquez à quoi correspond la contrainte d'intégrité.*

Réponse :

$$Id \rightarrow (Nom, Prenom, Sexe, DateNaissance, DateDeces)$$

La date de décès ne peut être antérieure à la date de naissance.

```
CREATE TABLE Genealogie.Parents (
  -- Typage des attributs
  Enfant integer NOT NULL,
  Mere integer NOT NULL,
  Pere integer NOT NULL,
  -- Clefs candidates
  PRIMARY KEY (Enfant),
```

```
  -- Clefs etrangeres
  FOREIGN KEY (Enfant) REFERENCES Genealogie.Personnes(Id),
  FOREIGN KEY (Mere) REFERENCES Genealogie.Personnes(Id),
  FOREIGN KEY (Pere) REFERENCES Genealogie.Personnes(Id),
  -- Contraintes d'integrite elementaire
  CHECK (Enfant <> Mere AND Enfant <> Pere AND Mere <> Pere)
);

-- PostgreSQL exige une fonction pour les contraintes d'integrite avec SELECT
CREATE FUNCTION Genealogie.AgeMerePereCorrects(integer, integer, integer)
  RETURNS boolean AS $$
SELECT EXISTS (
  SELECT *
  FROM
    (SELECT DateNaissance FROM Genealogie.Personnes WHERE $1 = Id) AS DateEnfant,
    (SELECT DateNaissance FROM Genealogie.Personnes WHERE $2 = Id) AS DateMere,
    (SELECT DateNaissance FROM Genealogie.Personnes WHERE $3 = Id) AS DatePere
  WHERE Extract(year FROM DateEnfant.DateNaissance) >
        Extract(year FROM DateMere.DateNaissance) + 10
    AND Extract(year FROM DateEnfant.DateNaissance) >
        Extract(year FROM DatePere.DateNaissance) + 10)
$$ LANGUAGE SQL;
-- Ajout de la contrainte par appel de la fonction
ALTER TABLE Genealogie.Parents ADD CONSTRAINT AgesParentsPossibles
  CHECK(Genealogie.AgeMerePereCorrects(Enfant, Mere, Pere) = TRUE);

-- PostgreSQL exige une fonction pour les contraintes d'integrite avec SELECT
CREATE FUNCTION Genealogie.SexeMerePereCorrects(integer, integer)
  RETURNS boolean AS $$
SELECT NOT EXISTS (
  SELECT *
  FROM Genealogie.Personnes
  WHERE ($1 = Id AND Sexe = 'M') OR ($2 = Id AND Sexe = 'F'))
$$ LANGUAGE SQL;
-- Ajout de la contrainte par appel de la fonction
ALTER TABLE Genealogie.Parents ADD CONSTRAINT SexesParentsPossibles
  CHECK(Genealogie.SexeMerePereCorrects(Mere, Pere) = TRUE);
```

Question 1.2 (1 point) *Pour la relation* `Parents`*, donnez la seule dépendance fonctionnelle déclarée, puis expliquez à quoi correspondent les trois contraintes d'intégrité.*

Réponse :

$$Enfant \rightarrow (Mere, Pere)$$

- *Pour chaque triplet, les identités de l'enfant, de la mère et du père sont toutes différentes.*
- *Les parents doivent avoir 10 ans au moment de la naissance de leurs enfants.*
- *Le père doit être de sexe masculin, et la mère de sexe féminin.*

```
CREATE TABLE Genealogie.Mariages (
  -- Typage des attributs
  Femme serial NOT NULL,
  Mari serial NOT NULL,
  DateMariage date NOT NULL,
  DateDivorce date NOT NULL DEFAULT 'infinity',
  -- Clefs candidates
  PRIMARY KEY (Femme, Mari, DateMariage),
  UNIQUE (Femme, Mari, DateDivorce),
  -- Clefs etrangeres
  FOREIGN KEY (Femme) REFERENCES Genealogie.Personnes(Id),
  FOREIGN KEY (Mari) REFERENCES Genealogie.Personnes(Id),
  -- Contraintes d'integrite elementaire
```

```
  CHECK (Femme <> Mari),
  CHECK (DateMariage < DateDivorce)
);

-- PostgreSQL exige une fonction pour les contraintes d'integrite avec SELECT
CREATE FUNCTION Genealogie.DatesMariageCorrectes(integer, integer, date, date)
  RETURNS boolean AS $$
SELECT NOT EXISTS (
  SELECT *
  FROM Genealogie.Mariages
  WHERE ($1 = Femme AND $3 <= DateMariage AND $4 >= DateDivorce)
     OR ($2 = Mari AND $3 <= DateMariage AND $4 >= DateDivorce))
$$ LANGUAGE SQL;
-- Ajout de la contrainte par appel de la fonction
ALTER TABLE Genealogie.Mariages ADD CONSTRAINT DatesPossibles
  CHECK(Genealogie.DatesMariageCorrectes(Femme, Mari, DateMariage, DateDivorce) = TRUE);

-- PostgreSQL exige une fonction pour les contraintes d'integrite avec SELECT
CREATE FUNCTION Genealogie.SexeFemmeMariCorrects(integer, integer)
  RETURNS boolean AS $$
SELECT NOT EXISTS (
  SELECT *
  FROM Genealogie.Personnes
  WHERE ($1 = Id AND Sexe = 'M') OR ($2 = Id AND Sexe = 'F'))
$$ LANGUAGE SQL;
-- Ajout de la contrainte par appel de la fonction
ALTER TABLE Genealogie.Mariages ADD CONSTRAINT SexesCouplePossibles
  CHECK(Genealogie.SexeFemmeMariCorrects(Femme, Mari) = TRUE);
```

Question 1.3 (1 point) *Pour la relation* `Mariages`*, donnez les seules dépendances fonctionnelles déclarées, puis expliquez à quoi correspondent les quatre contraintes d'intégrité.*

Réponse :

$$\begin{aligned}(Femme, Mari, DateMariage) &\rightarrow DateDivorce\\ (Femme, Mari, DateDivorce) &\rightarrow DateMariage\end{aligned}$$

- *La femme et le mari ne font pas qu'un.*
- *Ils doivent se marier avant de divorcer.*
- *Une femme ne peut pas être mariée à deux hommes en même temps.*
- *Un homme ne peut pas être marié à deux femmes en même temps.*
- *Le mari est de sexe masculin et la femme de sexe féminin.*

Remarque complémentaire non demandée : *Les contraintes sur la monogamie ajaoute en fait 4 nouvelles DFs.*

$$\begin{aligned}(Femme, DateMariage) &\rightarrow (Mari, DateDivorce)\\ (Femme, DateDivorce) &\rightarrow (Mari, DateMariage)\\ (Mari, DateMariage) &\rightarrow (Femme, DateDivorce)\\ (Mari, DateDivorce) &\rightarrow (Femme, DateMariage)\end{aligned}$$

Question 1.4 (1 point) *En tenant compte uniquement des dépendances fonctionnelles que vous avez listées dans les réponses précédentes, dites si les relations* `Personnes`*,* `Parents` *et* `Mariages` *sont 3NF et/ou BCNF.*

Réponse :

	3NF	*BCNF*	*Justification*
Personnes	*OUI*	*OUI*	$Id \rightarrow (Nom, Prenom, Sexe, DateNaissance, DateDeces)$ *est la seule DF irréductible à gauche, donc BCNF* *BNCF* $\Rightarrow$ *3NF*
Parents	*OUI*	*OUI*	$Enfant \rightarrow (Mere, Pere)$ *est la seule DF irréductible à gauche, donc BCNF* *BNCF* $\Rightarrow$ *3NF*
Mariages	*OUI*	*OUI*	*Cas 1 : Les 2 DFs déclarées seules* *Les 2 clefs candidates sont les membres gauches des 2 DFs, donc BCNF* *Cas 2 : Les 2 DFs plus les 4 DFs* *Les 2 premières ne sont pas irréductibles, donc 4 DFs* *Les 4 clefs candidates sont les membres gauches des 4 DFs, donc BCNF* *BNCF* $\Rightarrow$ *3NF*

Question 1.5 (1 point) *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les identifiants des* `Enfant` *ayant pour mère* 7 *et pour père* 4.

Réponse : $R = \pi[Enfant](\sigma[Mere = 7 \wedge Pere = 4](Parents))$

```
SELECT Enfant
FROM   Genealogie.Parents
WHERE  Genealogie.Parents.Mere    = 7
  AND  Genealogie.Parents.Pere    = 4;
```

Question 1.6 (1,5 point) *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les noms, prénoms et date de naissance des enfants de* 4. *La requête listera les réponses par ordre croissant des dates de naissance.*

Réponse : *Réponse non unique.*
$R = \pi[Nom, Prenom, DateNaissance](\sigma[Id = Enfant](Personnes \times \pi[Enfant](\sigma[Mere = 4 \vee Pere = 4](Parents))))$

```
SELECT Nom, Prenom, DateNaissance
FROM   Genealogie.Personnes,
       (SELECT Enfant
        FROM   Genealogie.Parents
        WHERE  (Genealogie.Parents.Mere = 4 OR Genealogie.Parents.Pere = 4))
       AS Enfants
WHERE  Id = Enfant
ORDER BY DateNaissance;
```

Question 1.7 (1,5 point) *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les noms, prénoms et date de naissance des demi-frères et des demi-soeurs de* 4. *La requête listera les réponses par ordre croissant des dates de naissance.*

Réponse : *Réponse non unique.*

$$\begin{aligned} R = \; & \pi[Nom,\ Prenom,\ DateNaissance](\sigma[Id = Enfant](Personnes \times (\\ & \pi[Mere.Enfant](\sigma[Parents.Enfant = 4 \wedge Parents.Mere = Mere.Mere](Parents \times \alpha(Parents : Mere))) \\ & \cup \\ & \pi[Pere.Enfant](\sigma[Parents.Enfant = 4 \wedge Parents.Pere = Pere.Pere](Parents \times \alpha(Parents : Pere)))) \end{aligned}$$

```
SELECT Nom, Prenom, DateNaissance
FROM   Genealogie.Personnes,
       (SELECT Mere.Enfant
        FROM   Genealogie.Parents AS EnfantsM, Genealogie.Parents AS Mere
        WHERE  EnfantsM.Enfant = 4
          AND  EnfantsM.Mere   = Mere.Mere
       UNION
        SELECT Pere.Enfant
        FROM   Genealogie.Parents AS EnfantsP, Genealogie.Parents AS Pere
        WHERE  EnfantsP.Enfant = 4
```

```
          AND  EnfantsP.Pere   = Pere.Pere)
        AS Enfants
WHERE  Id = Enfant
ORDER BY DateNaissance;
```

Question 1.8 (1,5 point) *Écrire une requête SQL qui liste les noms et les prénoms des personnes qui se sont mariés plus de 4 fois.*

Réponse : *Réponse non unique.*

```
SELECT Nom, Prenom, Nombre
FROM   Genealogie.Personnes,
       (SELECT Femme AS Humain, Count(*) AS Nombre
        FROM   Genealogie.Mariages
        GROUP BY Femme
        HAVING Count(*) > 4
          UNION
        SELECT Mari AS Humain, Count(*) AS Nombre
        FROM   Genealogie.Mariages
        GROUP BY Mari
        HAVING Count(*) > 4
       ) AS Recidiviste
WHERE  Personnes.Id = Recidiviste.Humain;
```

Question 1.9 (1 point) *Lister les problèmes posés par la fusion des relations* `Personnes` *et* `Parents` *en une seule relation qui contiendrait l'union des attributs.*

Réponse :
C'est le problème de l'oeuf et de la poule. On ne peut pas ajouter une personne sans connaître ses parents.
Une solution consiste à accepter des attributs à valeurs indéfinies. Deux inconvénients : on sort du cadre relationnel, et les contraintes d'intégrité sont très difficiles à écrire.

Question 1.10 (1,5 point) *Une personne propose de scinder la relations* `Personnes` *en deux relations* `Hommes` *et* `Femmes` *ayant les mêmes listes d'attributs* (`Id, Nom, Prenom, DateNaissance, DateDeces`). *Donnez les avantages et les inconvénients de ce nouveau schéma par rapport au précédent.*

Réponse :

Avantages : *Certaines contraintes s'écrivent plus facilement car il n'est pas nécessaire de faire les tests sur le sexe d'une personne.*

Inconvénients : *Un homme et une femme peuvent avoir les mêmes identifiants, ce qui demande d'ajouter un attribut (lequel ?) dans la table parents qui n'est plus une relation, car il n'y a plus de clefs candidate possible. Une solution consiste à imposer que les identifiants soient différents (positifs pour les uns et négatifs pour les autres, ou bien pairs et impairs), ce qui revient à encoder l'information du sexe. Une autre solution consiste à faire deux relations parents : parents des garçons et parents des filles. Les requêtes deviennent compliquées à écrire.*

Exercice 2 (Normalisation (7 points))
Soit la relation `Concerts (Salle, Categorie, NbPlaces, Jour, Orchestre, TypeMusique, Soliste)`*, vision simplifiée d'une gestion de concerts, et un ensemble irréductible de dépendances fonctionnelles :*
- $\{$`Salle`$\} \longrightarrow \{$`Categorie`$\}$ *: une salle détermine sa catégorie.*
- $\{$`Categorie`$\} \longrightarrow \{$`NbPlaces`$\}$ *: la catégorie d'une salle détermine son nombre de places.*
- $\{$`Soliste`$\} \longrightarrow \{$`TypeMusique`$\}$ *: un soliste ne joue que d'un seul type de musique.*
- $\{$`Salle, Jour`$\} \longrightarrow \{$`Orchestre`$\}$ *: par jour, une salle n'est occupée que par un seul orchestre.*
- $\{$`Salle, Jour`$\} \longrightarrow \{$`Soliste`$\}$ *: par jour, une salle n'est occupée que par un seul soliste.*
- $\{$`Orchestre, TypeMusique`$\} \longrightarrow \{$`Soliste`$\}$ *: pour chaque type de musique, un orchestre possède son soliste.*
- $\{$`Orchestre, Jour`$\} \longrightarrow \{$`Salle`$\}$ *: un orchestre ne joue pas dans deux salles différentes le même jour.*
- $\{$`Soliste, Jour`$\} \longrightarrow \{$`Salle`$\}$ *: un soliste ne joue pas dans deux salles différentes le même jour.*

Salle	*Categorie*	*NbPlaces*	*Jour*	*Orchestre*	*TypeMusique*	*Soliste*
Playel	*Theatre*	*1000*	*27-03-2008*	*Velvet*	*Rock*	*Lou Reed*
Parc des Princes	*stade*	*40000*	*26-03-2008*	*E Street Band*	*Rock*	*Bruce Springsteen*

Question 2.1 (1 point) *Donnez toutes les clefs candidates de la relation* `Concerts`.

Réponse : *Les dépendances fonctionnelles donnent :*
- $C_1 = \{$`Salle, Jour`$\}$
- $C_2 = \{$`Soliste, Jour`$\}$
- $C_3 = \{$`Orchestre, Jour`$\}$

Question 2.2 (1 point) *Même si l'on suppose qu'il n'y a aucun doublon dans* `Concerts`*, justifiez pourquoi la relation* `Concerts` *n'est pas en troisième forme normale.*

Réponse : *Une seule des explications suivantes est suffisante (liste non exhaustive).*

Non 2NF : *La clef* $\{$`Salle, Jour`$\}$ *contient* $\{$`Salle`$\}$ *qui détermine* $\{$`Categorie`$\}$.

Non 3NF : *La clef* $\{$`Salle, Jour`$\}$ *et* $\{$`Categorie`$\} \longrightarrow \{$`NbPlaces`$\}$.

Non 3NF : *La clef* $\{$`Salle, Jour`$\}$ *et* $\{$`Soliste`$\} \longrightarrow \{$`TypeMusique`$\}$.

Question 2.3 (2 points) *Appliquez un algorithme (ou une technique) de normalisation pour obtenir une décomposition, sans perte d'information, de la relation* `Concerts` *en un ensemble de relations au moins en troisième forme normale. Vous n'écrirez sur la copie que les nouvelles relations et les dépendances fonctionnelles qui sont à la base des projections effectuées.*

Réponse : *Décomposition en BCNF :*

1. `Categories (Categorie, NbPlaces)` *qui provient de* $\{$`Categorie`$\} \longrightarrow \{$`NbPlaces`$\}$ *(BCNF)*
2. `Salles (Salle, Categorie)` *qui provient de* $\{$`Salle`$\} \longrightarrow \{$`Categorie`$\}$ *(BCNF)*
3. `Solistes (Soliste, TypeMusique)` *qui provient de* $\{$`Soliste`$\} \longrightarrow \{$`TypeMusique`$\}$ *(BCNF) mais la dépendance* $\{$`Orchestre, TypeMusique`$\} \longrightarrow \{$`Soliste`$\}$ *devient une contrainte inter relations.*
4. `JoursConcerts (Salle, Jour, Orchestre, Soliste)` *(BCNF)*

Décomposition en 3NF :

1. `Categories (Categorie, NbPlaces)` *qui provient de* $\{$`Categorie`$\} \longrightarrow \{$`NbPlaces`$\}$ *(BCNF)*
2. `Salles (Salle, Categorie)` *qui provient de* $\{$`Salle`$\} \longrightarrow \{$`Categorie`$\}$ *(BCNF)*
3. `JoursConcerts (Salle, Jour, Orchestre, Soliste)` *qui provient de* $\{$`Salle, Jour`$\} \longrightarrow \{$`Orchestre, Soliste`$\}$ *(BCNF)*
4. `TypesConcerts (Orchestre, TypeMusique, Soliste)` *qui provient de* $\{$`Orchestre, TypeMusique`$\} \longrightarrow \{$`Soliste`$\}$ *et qui contient* $\{$`Soliste`$\} \longrightarrow \{$`TypeMusique`$\}$. *(3NF et non BCNF)*

Question 2.4 (3 points) *Après avoir précisé si votre décomposition est en BCNF ou bien seulement en 3NF, répondez aux questions qui vous concernent.*

Votre décomposition est en BCNF *:*

1. *Indiquez la dépendance fonctionnelle que vous avez perdue.*
2. *En supposant que cette dépendance ne soit pas écrite sous forme d'une contrainte, donnez un ensemble de requêtes d'insertion pour vos relations, qui viole cette dépendance fonctionnelle.*

Votre décomposition est seulement en 3NF *:*

1. *Indiquez le problème de redondance qui subsiste.*
2. *Donnez un ensemble de requêtes d'insertion pour vos relations, suivi d'une requête de mise à jour qui nécessite que le SGBD modifie eventuellement plusieurs tuples, du fait de la redondance.*

Réponse :

Décomposition en BCNF *:*

1. $\{$`Orchestre, TypeMusique`$\} \longrightarrow \{$`Soliste`$\}$.
2. *Les insertions suivantes sont possibles :*

```
INSERT INTO Solistes VALUES (s1, t1);
INSERT INTO Solistes VALUES (s2, t1);
INSERT INTO JoursConcerts VALUES (salle1, '01-01-2011', orc1, s1);
INSERT INTO JoursConcerts VALUES (salle1, '02-01-2011', orc1, s2);
```

Votre décomposition est seulement en 3NF *:*

1. *L'information* (Soliste, TypeMusique) *est dupliquée.*

2. *La mise à jour suivante modifie plusieurs tuples.*

```
INSERT INTO TypesConcerts VALUES (orc1, t1, s1);
INSERT INTO TypesConcerts VALUES (orc2, t1, s1);
UPDATE TypesConcerts SET (TypeMusique = t2) WHERE Soliste = s1;
```

Exercice 3 (Reprise après une panne (4 points))
Nous supposons que le SGBD écrit dans un journal le début, les opérations et la fin de chaque transaction, et que chaque écriture est aussitôt sauvegardée sur disque. Nous supposons également que le SGBD utilise la technique des points de synchronisation : à intervalles réguliers le SGBD écrit dans le journal la liste des transactions en cours, puis aussitôt sauvegarde le journal et la base de données sur le disque dur.
Soit pc le dernier point de synchronisation réalisé à une date tc, et une panne du SGDB à la date $tf > tc$.

Question 3.1 (1 point) *Donner les objectifs de l'algorithme de reprise après la panne.*

Réponse : *L'objectif principal est la reprise du service. Cette reprise doit pénaliser le moins possible les applications clientes. Elle doit donc remettre la base dans un état cohérent vis à vis de l'information connue des clients, ainsi :*
- *Un client ne doit pas avoir à refaire une transaction terminée avant la panne.*
- *Un client doit avoir à refaire une transaction non terminée avant la panne.*

Question 3.2 (3 points) *Donner l'algorithme de reprise après la panne.*

Réponse : *(cf pages 53 et 54 du polycopie)*