Proposition de correction de l'examen du cours *Systèmes d'information*

S1934 – MINES ParisTech

9 juin 2016

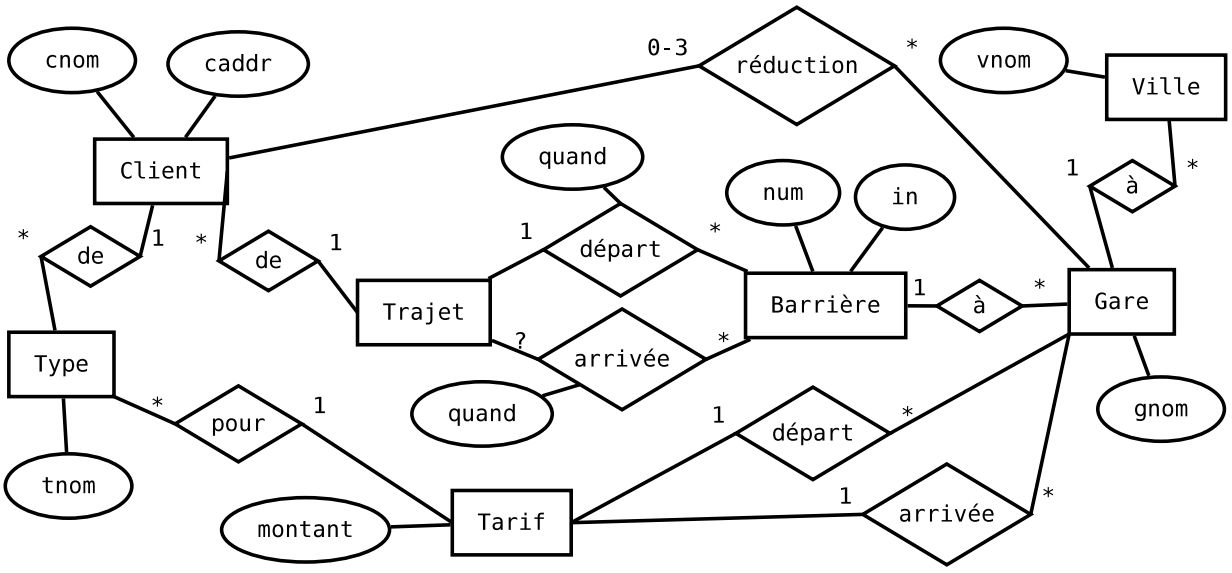
Bravo Calvin ! 20/20

1 Modélisation EA *6*/6

Construire un modèle EA pour représenter la situation suivante : la société d'autoroute *EU-roads* gère un réseau autoroutier à péage et propose un service de *Télé-péage*. Une gare de péage est composée de barrières identifiées, et est associée à une ville (*e.g.* barrière d'entrée numéro 2 de la Gare de péage Val-de-Loing associée à la ville de Nemours). Un abonné (moto, voiture, camion) possède un badge pour signaler son passage sans arrêt à une barrière. Lors d'un trajet sur le réseau, l'abonné signale son passage à une barrière d'entrée de l'autoroute, puis à une sortie. Le système permet de garder l'historique des trajets des abonnés. À chaque trajet (paire gare d'entrée - gare de sortie) et type de véhicule (moto, voiture ou camion) correspond un tarif. Un abonné peut avoir une liste de trois gares pour lesquelles il a une réduction de 20% sur les trajets qui entrent et sortent de ces gares.

Vous pouvez aussi ajouter quelques commentaires si vous les estimez nécessaires, ou discuter brièvement les motivations des choix de modélisation faits et les limites du modèle.

Voici une proposition de modèle EA pour EU-roads *:*



Il n'est pas prévu qu'un même client puisse avoir plusieurs abonnements, ce qui est une limitation discutable.

2 Traduction relationnelle *4/4*

À partir du modèle EA précédent, construisez un modèle relationnel. Vous prendrez soin de bien préciser les champs utiles et les contraintes pertinentes sur vos relations. Vous commenterez les contraintes que vous ne pourriez exprimer directement dans le modèle.

Vous pourrez utiliser les abréviations suivantes : **CT** pour `CREATE TABLE`, **I** pour `INTEGER`, **S** pour `SERIAL`, **T** pour `TEXT`, **D** pour `DATE`, **Ts** pour `TIMESTAMP`, **B** pour `BOOLEAN` **PK** pour `PRIMARY KEY`, **U** pour `UNIQUE`, **NN** pour `NOT NULL`, **R** pour `REFERENCES`...

```
CT Ville(vid SPK, nom TUNN);
CT Gare(gid SPK, nom TUNN, vid INNR Ville);
CT Barriere(bid SPK, entree BNN, num INN, gid INNR Gare, U(entree, num, gid));
CT Type(tid SPK, type TUNN);
CT Client(cid SPK, nom TNN, addr TNN, U(nom, addr)?, tid INNR Type,
          reductions I[] CHECK(array_length(reductions) <= 3 AND
            -- vérification des références à la main...
            array_length(reductions) =
              (SELECT COUNT(DISTINCT gid) FROM Gare WHERE gid IN reductions)));
CT Trajet(jid SPK, cid INNR Client, entree INNR Barriere, its TsNN, U(cid, its),
          sortie IR Barriere, ots TsNN,
          CHECK(sortie IS NULL AND ots IS NULL OR
                sortie IS NOT NULL AND ots IS NOT NULL));
CT Tarif(tid SPK, gin INNR Gare, gout INNR Gare, tid INNR Type,
         U(gin, gout, tid));
```

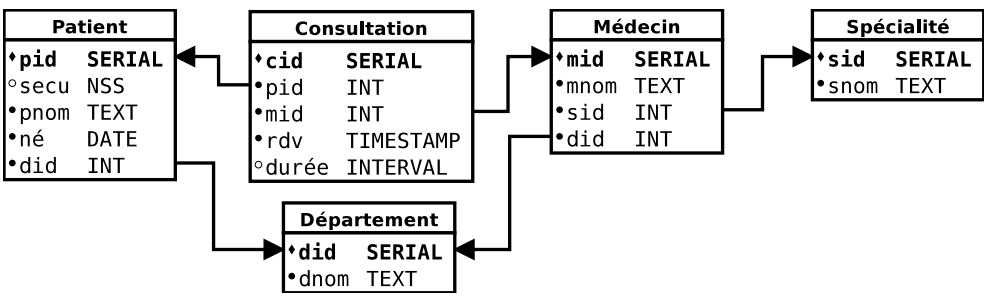
Un trajet concerne une barrière d'entrée vers une de sortie, mais cette contrainte n'est pas imposée.

La contrainte d'unicité sur le Client mériterait d'être mieux pensée.

Le tableau de 3 références pour les réductions est assez discutable.

3 Requêtes *6/6*

On considère le modèle relationnel suivant, qui représente des patients qui consultent des médecins de certaines spécialités (les attributs en gras sont les clefs primaires, les points noirs désignent des attributs `NOT NULL`, les flèches sont des clefs étrangères) :



Proposez **une** requête (au moins sa structure, la syntaxe exacte n'est pas essentielle) pour chacune des questions suivantes :

1. Quelles sont les consultations du patient *Calvin* dans la spécialité *ORL* en *juin 2016* ?

```
SELECT c.*
FROM Patient AS p
JOIN Consultation AS c USING (pid)
JOIN Médecin AS m USING (mid)
JOIN Spécialité AS s USING (sid)
WHERE p.pnom = 'Calvin'
  AND c.rdv BETWEEN DATE '2016-06-01' AND DATE '2016-06-30' -- relation d'ordre
  AND s.snom = 'ORL';
```

   Pour cette requête uniquement, suggérez les indexes (hors clefs primaires) potentiellement utiles pour en améliorer les performances.

```
-- en partant du nom du Patient - le plus sélectif ?
CREATE INDEX patient_pnom ON Patient(pnom);              -- pnom -> pid
CREATE INDEX consultation_pid ON Consultation(pid);      -- pid  -> cid
-- en partant de la date de la Consultation
CREATE INDEX consultation_rdv ON Consultation(rdv);      -- rdv  -> cid
-- en partant du nom de la Spécialité
CREATE INDEX specialité_snom ON Specialité(snom);        -- snom -> sid
CREATE INDEX médecin_sid ON Médecin(sid);                -- sid  -> mid
CREATE INDEX consultation_mid ON Consultation(mid);      -- mid  -> cid
```

2. Combien de consultations de plus d'une heure **et** combien de moins d'une heure ont été réalisées par le médecin *Hobbes* ?

```
SELECT
  COUNT(*) FILTER (WHERE c.durée < INTERVAL '1 h'),
  COUNT(*) FILTER (WHERE c.durée >= INTERVAL '1 h')
FROM Consultation AS c
JOIN Médecin AS m USING (mid)
WHERE m.mnom = 'Hobbes';
```

3. Quels médecins du département du *Loiret* n'ont pas été consultés en 2015 ?

```
WITH MédecinDuLoiret AS (
  SELECT m.*
  FROM Médecin AS m
  JOIN Département AS d USING (did)
  WHERE d.dnom = 'Loiret')
-- tous les médecins du Loiret
SELECT m.*
FROM MédecinDuLoiret AS m
EXCEPT
-- moins ceux ayant travaillé en 2015
SELECT DISTINCT m.*
FROM MédecinDuLoiret AS m
JOIN Consultation AS c USING (mid)
WHERE c.rdv BETWEEN DATE '2015-01-01' AND DATE '2015-12-31';
```

4. Quel est le nombre de patients par médecin dans chaque département classé par ordre alphabétique ?

```
SELECT ppd.dnom, ppd.nb / mpd.nb
FROM -- # patients par département
  (SELECT d.dnom, COUNT(*) AS nb
   FROM Patient AS p
   JOIN Département AS d USING (did)
   GROUP BY dnom) AS ppd
JOIN -- # médecins par département
  (SELECT d.dnom, COUNT(*) AS nb
   FROM Médecin AS m
   JOIN Département AS d USING (did)
   GROUP BY dnom) AS mpd
     USING (dnom)
ORDER BY dnom;
```

5. Quelles paires de patients (donner leurs `pid`) ont consulté au moins trois médecins en commun ?

```sql
WITH MédecinCommun AS (
  SELECT DISTINCT p1.pid AS pid1, p2.pid AS pid2, c1.mid AS mid
  FROM Patient AS p1
  JOIN Consultation AS c1 USING (pid)
  JOIN Consultation AS c2 USING (mid)
  JOIN Patient AS p2 ON (p2.pid=c2.pid)
  WHERE p1.pid < p2.pid -- évite les doublons
)
SELECT DISTINCT pid1, pid2
FROM MédecinCommun
GROUP BY pid1, pid2
HAVING COUNT(*) >= 3;
```

4 Questions de cours *4/4*

Choisissez un thème parmi les deux tirés aléatoirement en début d'examen dans la liste *PostgreSQL, Relationnel, Transaction, MVCC, Optimisation, Droits, PL/pgSQL, JDBC, SIG, Systèmes distribués*, et expliquez en moins de 100 mots ce que vous en avez retenu.

***Thème** : ACID (Advanced Chanting In Databases)*

Ce thème est très passionnant et j'ai appris plein de choses super intéressantes qui éclairent parfaitement le fonctionnement de cet aspect des bases de données à la fois théorique et pratique, et me permettent de bien comprendre les caractéristiques générales et particulières des multiples facettes de ce thème dans une perspective de mise en application concrète du modèle relationnel sur des problèmes réels, tout en gardant, au delà du simple niveau fonctionnel, une maîtrise des aspects opérationnels sur PostgreSQL qui gère le stockage de tables pour un prix compatible avec toutes les bourses, une bonne nouvelle pour nos budgets !

Citez les noms de trois scientifiques ayant obtenu le prix Turing pour leurs travaux de recherche sur les bases de données.

Il y a 4 prix Turing liés aux recherches sur les bases de données :

1. *Charles William (Charlie) Bachman (USA 1924-) – 1973*
2. *Edgard Franck (Ted) Codd (UK 1923-2003) – 1981*
3. *James Nicholas (Jim) Gray (USA 1944-2007/2012) – 1998*
4. *Michael Ralph Stonebraker (USA 1943-) – 2014*

Distribution des 28 notes : min 8,6 moyenne 14,6 médian 15,3 max 20

Document composé avec LaTeX, révision 3503.