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

Fabien Coelho et Claire Medrala – MINES Paris

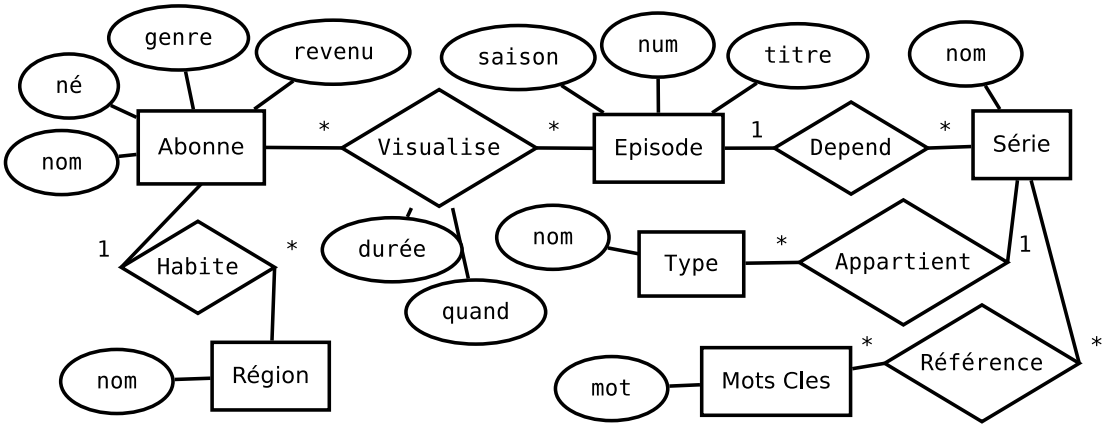
Janvier 2021

Bravo Hobbes ! 20/20

1 Modélisation entité-association *5*/5

une plateforme télévisuelle propose des épisodes de séries de différents types. Chaque série est aussi caractérisée par un ensemble de mots clés. Les abonnés sont caractérisés par leur genre, leur année de naissance, leur région, leur niveau de revenu annuel. Le système garde trace de toutes leurs visualisations, en particulier la durée de visionnage afin de savoir si ils ont regardé un épisode en entier, pour aider à mieux cibler les nouvelles propositions du moteur de recommandations.

Voici une proposition de modèle E/A pour cette situation :



2 Traduction relationnelle *3*/3

À partir du modèle E/A précédent, construisez un modèle relationnel.

```
CT Region(rid SPK, rnom TUNN);
CT MotClé(mid SPK, mot TUNN);
CT Type(tid SPK, tnom TUNN);
CT Série(sid SPK, snom TUNN, tid INNR Type);
CT MotCleSérie(sid INNR Série, mid INNR MotClé, PK(sid, mid));
CT Abonné(aid SPK, nom TNN, genre C(1)NN, né DNN, rev FNN,
          rid INNR Région, U(nom,genre,né));
CT Épisode(eid SPK, saison INN, num INN, U(saison, num),
           sid INNR Série, titre TNN, U(sid, titre));
CT Visualise(vid SPK, aid INNR Abonné, eid INNR Épisode, quand TsNN,
             U(aid, eid, quand), durée In);
```

La durée est `NULL` en début de visionnage.

3 Requêtes *8/8*

On considère un modèle relationnel concernant un système de location de vélos de type *Vlib*.

1. Quels vélos, identifiés par leurs numéros, ont été empruntés par l'utilisateur *Calvin* le *11 décembre 2020* au parking *Pont Neuf* ?

```sql
SELECT DISTINCT v.vnum
FROM utilisateur AS u
JOIN emprunt AS e USING (uid)
JOIN estgare AS g ON (g.gid = e.edep)
JOIN borne AS b USING (bid)
JOIN parking AS p USING (pid)
JOIN velo AS v USING (vid)
WHERE u.unom = 'Calvin'
  AND p.pnom = 'Pont Neuf'
  AND g.gdep BETWEEN '2020-12-11 00:00:00' AND '2020-12-11 23:59:59';
```

   Suggestions d'index :

```sql
-- en partant de Utilisateur :
-- Utilisateur(unom) : existe déjà car UNIQUE
CREATE INDEX emprunt_uid ON Emprunt(uid);
-- -> EstGaré(gid) -> Borne(bid) -> Parking(pid) -> Vélo(vid)
-- en partant Parking :
-- Parking(pnom)
CREATE INDEX borne_pid ON Borne(pid);
CREATE INDEX estgaré_bid ON EstGare(bid);
CREATE INDEX emprunt_edep ON Emprunt(edep);
-- Vélo(vid), Utilisateur(pid) : PK
-- en partant de EstGaré :
CREATE INDEX estgare_gdep ON EstGare(gdep);
-- Borne(bid) -> Parking(pid) -> emprunt_edep -> Utilisateur(pid) -> Vélo(vid)
```

2. Quels vélos, identifiés par leurs numéros, ont été garés dans au moins quatre parkings différents, par ordre de leurs numéros ?

```sql
SELECT vnum
FROM Velo AS v
JOIN EstGare AS g USING (vid)
JOIN Borne AS b USING (bid)
JOIN Parking AS p USING (pid)
GROUP BY 1
HAVING COUNT(DISTINCT p.pid) >= 4
ORDER BY 1;
```

3. Compter le nombre de vélos actuellement garés et le nombre de vélos actuellement empruntés, sous forme de deux tuples, y compris si une réponse est zéro.

```sql
-- vélos garés
SELECT 'Garés' AS etat, COUNT(*) FILTER (WHERE gdep IS NULL) AS "#"
FROM EstGare
UNION
SELECT 'Empruntés', COUNT(*) FILTER (WHERE earr IS NULL)
FROM Emprunt;
```

4. Pour *tous* les utilisateurs, la distance parcourue, et leur rang dans l'ordre alphabétique de leurs noms, par ordre des distances décroissantes, puis des noms.

```sql
SELECT unom, SUM(edist), RANK() OVER (ORDER BY enom)
FROM Utilisateur
LEFT JOIN Emprunt USING (uid)
GROUP BY 1
ORDER BY 2 DESC, 1 ASC;
```

5. Pour *tous* les vélos, donner le temps total et le nombre d'emprunts, par ordre décroissant des temps, puis par ordre de leurs numéros.

```sql
SELECT vnum, SUM(a.garr - d.gdep) AS durée, COUNT(e.eid) AS nb
FROM Velo AS v
JOIN EstGare AS d USING (vid) -- départ du parking
JOIN EstGare AS a USING (vid) -- arrivé au parking
JOIN Emprunt AS e ON (d.gid = e.edep AND a.gid = e.earr)
GROUP BY 1
UNION
-- vélo jamais garé
SELECT vnum, INTERVAL '0', 0
FROM Velo
LEFT JOIN EstGare USING (vid)
WHERE gid IS NULL
UNION
-- vélo garé mais jamais emprunté
SELECT vnum, INTERVAL '0', 0
FROM Velo
WHERE vid NOT IN (SELECT vid FROM EstGare JOIN Emprunt ON (edep = gid))
ORDER BY 2 DESC, 1 ASC;
```

6. Redondance…
   — Le modèle présenté comporte au moins une redondance. Laquelle ?
   Le vélo a une distance parcourue, et chaque emprunt qui concerne un vélo via le EstGaré référencé a aussi une distance parcourue. On peut penser que la première information est la somme des autres.
   Autre contrainte : un emprunt concerne un vélo garé au départ puis à l'arrivée. Il faut que ce soit le même vélo !
   — Si c'est volontaire, quelle raison a pu pousser le concepteur à cela ?
   L'information est peut-être souvent requise et longue à (re)calculer.
   — Afficher les objets pour lesquels l'écart entre les deux informations redondantes est supérieur à 5%. On prendra garde à traiter correctement les cas particuliers.

```sql
SELECT vnum AS velo,
  -- la somme de NULL donne zéro
  CASE WHEN vdist > 0 THEN SUM(edist) / vdist - 1.0 ELSE NULL END AS taux
FROM Velo AS v
JOIN EstGare AS g USING (vid)
JOIN Emprunt AS e ON (e.edep = g.gid)
WHERE edist IS NOT NULL
GROUP BY vnum, vdist
HAVING CASE WHEN vdist > 0 THEN ABS(SUM(edist) / vdist - 1.0) ELSE NULL END > 0.05;
```

4 Questions de cours *4*/4

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

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

Ce thème est très passionnant, 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 transversale de mise en application concrète du modèle relationnel sur des problèmes réels, tout en gardant, au delà du simple niveau fonctionnel, une maîtrise généraliste des aspects opérationnels sur Postgres qui gère le stockage à 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 1924-2017) – 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*

Document composé avec LaTeX.