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

Fabien Coelho et Claire Medrala – MINES ParisTech

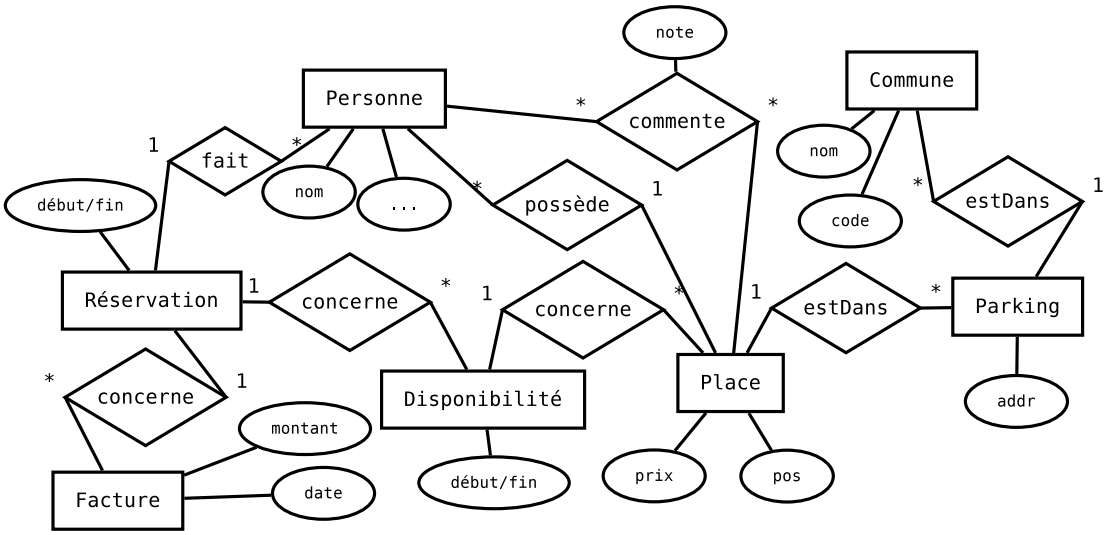
Mai 2019

***Bravo Hobbes ! 20/20***

## 1 Modélisation entité-association *6*/6

Une start-up investit le marché de la location de places de parkings. Des propriétaires proposent des places à louer pour lesquelles ils ouvrent des créneaux de disponibilité. Des automobilistes recherchent des places selon leur prix, leur localisation (commune) et leur disponibilité. Ils peuvent réserver des places pour des périodes de temps sur lesquelles elles sont disponibles. Ils sont ensuite facturés mensuellement pour leurs réservations. Ils notent et commentent les places utilisées.

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




*Les automobilistes et loueurs sont des personnes.*

*Les dates de réservations doivent être compatibles avec les disponibilités auxquelles elles se rattachent, et être compatibles entre elles (sans intersections).*

*Les dates de disponibilités doivent également être compatibles entre elles.*

## 2 Traduction relationnelle *4*/4

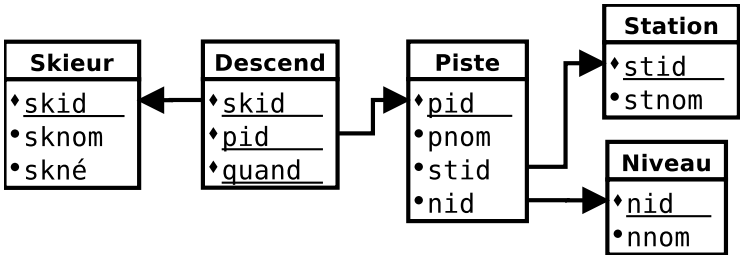
À partir du modèle E/A précédent, construisez un modèle relationnel. Vous prendrez soin de bien préciser les champs utiles et les contraintes pertinentes sur vos relations. Vous commenterez les contraintes que vous ne pourriez exprimer directement dans le modèle.

```
CT Commune(cid SPK, code TUNN, nom TUNN);
CT Personne(pid SPK, nom TNN, ..., U(nom, ...));
CT Parking(pkid SPK, addr TNN, cid INNR Commune, ..., U(addr, cid));
CT Place(plid SPK, pos TNN, plid INNR Parking,
    prix FNN, pid INN R Personne, ..., U(pos, plid));
CT Commentaire(cmid SPK, plid INNR Place, pid INNR Personne,
    note INN, ..., U(plid, pid));
CT Facture(fid SPK, montant FNN, ...);
CT Disponibilité(did SPK, plid INNR Place, début Ts NN, fin Ts NN,
    U(plid, début), CHECK (début < fin ));
CT Réservation(rid SPK, pid INNR Personne, did INNR Disponibilité,
    fid INNR Facture, début Ts NN, fin Ts NN, U(pid, plid, quand),
     CHECK(début < fin));
```

*L'unicité de certaines relations n'est pas évidente avec les attributs présents.*

# 3 Requêtes *6*/6

On considère le modèle relationnel suivant, qui représente les données de stations de ski qui collectent les pistes de différents niveaux descendues par les skieurs (les attributs soulignés sont les clefs primaires, les points noirs désignent des attributs `NOT NULL`, les flèches sont des clefs étrangères) :




1. Quels sont les noms des skieurs nés en *2012* qui ont descendu des pistes *bleues* de la station *Les Arcs* en *2019*, par ordre alphabétique ?

```
-- un skieur n'apparaît qu'une fois
SELECT DISTINCT sk.sknom
-- jointure de toutes les tables
FROM Skieur AS sk
JOIN Descend AS d USING (skid)
JOIN Piste AS p USING (pid)
JOIN Station AS st USING (stid)
JOIN Niveau AS n USING (nid)
-- filtrage des conditions
WHERE EXTRACT(YEAR FROM sk.skné) = 2012
  AND EXTRACT(YEAR FROM d.quand) = 2019
  AND n.nnom = 'Bleu'
  AND st.stnom = 'Les Arcs'
ORDER BY 1;
```

```
-- un skieur n'apparaît qu'une fois
SELECT DISTINCT sk.sknom
-- jointure de toutes les tables
FROM Skieur AS sk
JOIN Descend AS d USING (skid)
JOIN Piste AS p USING (pid)
JOIN Station AS st USING (stid)
JOIN Niveau AS n USING (nid)
-- filtrage des conditions
WHERE -- exploitation de la relation d'ordre...
      sk.skné BETWEEN '2012-01-01' AND '2012-12-31'
  AND d.quand BETWEEN '2019-01-01' AND '2019-12-31'
  AND n.nnom = 'Bleu'
  AND st.stnom = 'Les Arcs'
ORDER BY 1;
```

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

```
-- à partir de la station
CREATE INDEX station_stnom ON Station(stnom);
CREATE INDEX piste_stid ON Piste(stid);
CREATE INDEX descend_pid ON Descend(pid);
-- à partir du niveau
CREATE INDEX niveau_nnom ON Niveau(nnom);
CREATE INDEX piste_nid ON Piste(nid);
-- puis Descend(pid) comme ci-dessus
-- à partir des descentes
CREATE INDEX descend_quand ON Descend(quand);
-- à partir du skieur
CREATE INDEX skieur_né ON Skieur(skné);
```

2. Que est le nom du skieur qui a descendu le plus de pistes différentes (toutes stations confondues) en *2018* ?

```
SELECT s.sknom
-- les descentes des skieurs
FROM Skieur AS s
JOIN Descend AS d USING (skid)
-- de l'année
WHERE EXTRACT(YEAR FROM d.quand) = 2018
GROUP BY s.skid
-- tri par nombre de pistes différentes
ORDER BY COUNT(DISTINCT pid) DESC
LIMIT 1;
```

```
SELECT s.sknom
-- les descentes des skieurs
FROM Skieur AS s
JOIN Descend AS d USING (skid)
-- de la période
WHERE d.quand BETWEEN '2018-01-01' AND '2018-12-31'
GROUP BY s.skid
-- tri par nombre de pistes différentes
ORDER BY COUNT(DISTINCT pid) DESC
LIMIT 1;
```

```
-- avec une sous-requête réutilisée
WITH skieur_npistes(year, skid, nb) AS (
  SELECT EXTRACT(YEAR FROM quand) AS y, skid, COUNT(DISTINCT pid)
  FROM Descend
  GROUP BY y, skid)
SELECT sknom
FROM Skieur
JOIN skieur_npistes AS sn USING (skid)
WHERE sn.year = 2018
  AND sn.nb = (SELECT MAX(nb) FROM skieur_npistes WHERE year = 2018)
```

3. Quels sont les noms des stations sans piste *verte* ?

```
-- avec EXCEPT
-- toutes les stations
SELECT s.stnom
  FROM Station AS s
-- moins
EXCEPT
-- celles avec des pistes vertes
SELECT DISTINCT s.stnom
  FROM Station AS s
  JOIN Piste AS p USING (stid)
  JOIN Niveau AS n USING (nid)
  WHERE n.nnom = 'Vert';
```

```
-- avec LATERAL (bof...) et LEFT JOIN
SELECT st.stnom
FROM Station AS st
CROSS JOIN LATERAL (
  -- compte ses pistes vertes, y compris 0
  SELECT COUNT(p.pid)
  FROM Niveau AS n
  LEFT JOIN Piste AS p USING (nid)
  WHERE n.nnom = 'Vert'
    AND p.stid = st.stid
) AS c(nvert)
WHERE c.nvert = 0;
```

```
-- avec une sous-requête
SELECT s.stnom
FROM Station AS s
WHERE s.stnom NOT IN (
  -- pas celles avec des pistes vertes
  SELECT DISTINCT s.stnom
  FROM Station AS s
  JOIN Piste AS p USING (stid)
  JOIN Niveau AS n USING (nid)
  WHERE n.nnom = 'Vert');
```

4. Quel est le nom du skieur ayant skié le plus jeune ?

```sql
SELECT s.sknom
-- les skieurs et leurs descentes
FROM Skieur AS s
JOIN Descend AS d USING (skid)
-- ordonné par leur âge lors de cette descente
ORDER BY (d.quand - s.skné) ASC
-- on garde le premier, même si égalités
LIMIT 1;
```

```sql
-- avec une sous-requête
WITH début_skieur(sknom, âge) AS
 (SELECT s.sknom, MIN(d.quand - s.skné)
  FROM Skieur AS s
  JOIN Descend AS d USING (skid)
  GROUP BY sknom)
SELECT sknom FROM début_skieur
WHERE âge = (SELECT MIN(âge) FROM début_skieur);
```

5. Pour chaque nom de station, quelle est le nom de sa piste la plus fréquentée en *2018* ? La fréquentation est le nombre de descentes.

```sql
-- avec LATERAL (bof...)
SELECT st.stnom, p.pi
-- pour chaque station
FROM Station AS st
-- ... sa piste la plus fréquentée en 2018
CROSS JOIN LATERAL (
  SELECT p.pnom, COUNT(*)
  FROM Piste AS p
  JOIN Descend AS d USING (pid)
  WHERE p.stid = st.stid
    AND d.quand BETWEEN '2018-01-01'
                    AND '2018-12-31'
  GROUP BY 1
  ORDER BY 2 DESC, 1
  -- en cas d'égalité, une seule piste...
  LIMIT 1) AS p(pi, nb)
ORDER BY p.nb DESC;
```

```sql
-- avec OVER
WITH ranks(st, pi, nb, rk) AS
 (SELECT
    s.stnom, p.pnom,
    -- nombre total de descentes
    COUNT(*),
    -- rang du nombre de descente par station
    RANK() OVER (PARTITION BY s.stnom
                 ORDER BY COUNT(*) DESC)
  -- les descentes de chaque piste en 2018
  FROM Station AS s
  JOIN Piste AS p USING (stid)
  JOIN Descend AS d USING (pid)
  WHERE d.quand BETWEEN '2018-01-01'
                    AND '2018-12-31'
  GROUP BY s.stid, p.pid)
-- on garde le premier rang de chaque station,
-- plusieurs réponses si fréquences égales
SELECT st, pi FROM ranks WHERE rk = 1
ORDER BY nb DESC;
```

```sql
-- avec des sous-requêtes combinées
WITH
  -- fréquentation annuelle des pistes
  piste_freq(year, pid, nb) AS (
    SELECT EXTRACT(YEAR FROM quand) AS y, pid, COUNT(*)
    FROM Descend
    GROUP BY y, pid),
  -- meilleure fréquentation annuelle pour chaque station
  station_max(year, stid, nb) AS (
    SELECT year, stid, MAX(nb)
    FROM piste_freq
    JOIN Piste USING (pid)
    GROUP BY year, stid)
SELECT stnom, pnom
FROM station_max AS sm
JOIN Piste AS p USING (stid)
-- on retrouve la piste correspondant au max
JOIN piste_freq AS pf USING (year, pid, nb)
JOIN Station AS s USING (stid)
WHERE sm.year = 2018
ORDER BY nb DESC;
```

# 4 Questions de cours *4* /4

Choisissez un thème parmi les deux tirés aléatoirement en début d'examen dans la liste *Postgres, Relationnel, Transaction, MVCC, Optimisation, Droits, PL/pgSQL, Injection SQL, Décisionnel, 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, 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, révision 3955.