Université de Lille

TP 2.8
Renormalisation à partir de données réelles

CORRECTION + PÉDAGOGIE

M2106 - BDD
DUT Info – N2
Philippe Mathieu
2019–2020

Objectifs

L'objectif de ce TP consiste à récupérer des données issues du monde réel, puis intégrer et manipuler ces données dans un SGBD

Exercice 1 : Récupérer les données (rappels du cours Système)

A titre d'exemple, nous allons travailler sur les données électorales des élections présidentielles françaises du 24 avril 2017 fournies en *open data* par la Plateforme ouverte des données publiques françaises.

Q1. Téléchargez le fichier PR17_BVot_T1_FE.txt avec un click droit sur le bouton `Télécharger` puis "Save link as"

Q2. Quand on récupère un fichier de données, l'encodage utilisé peut parfois poser problème. Votre terminal étant UTF-8 il est préférable de manipuler des fichiers encodés de la même manière.
La commande `file -i` permet de voir l'encodage utilisé
la commande `iconv -f iso-8859-1 -t utf-8 fichier1 > fichier2` permet de changer d'encodage
réencodez ce fichier en l'appelant `data.csv`

file : c'est -i sous Unix, -I sous macos)

```
iconv -f iso-8859-1 -t utf-8 PR17_BVot_T1_FE.txt > data.csv
```

Q3. Ce fichier particulièrement mal structuré, contient 98 colonnes. Affichez dans une fenêtre shell grande ouverte la **première** ligne afin de visualiser les entêtes de ce fichier (commande unix `head` : man head)

```
head -n1 data.csv
```

Vous constatez que chaque bureau de chaque commune française est sur une ligne et l'ensemble des candidats sont en colonnes. De nombreuses colonnes (nom,prénom,sexe) sont donc répétées plein de fois. par ailleurs certaines colonnes peuvent être obtenues par calcul sur les autres (tous les pourcentages)

Q4. La première ligne ne contenant que des entêtes, affichez uniquement la **seconde** ligne afin de voir un exemple de données.

```
head -n2 data.csv | tail -n1
```

Q5. La ligne 69085 a quelque chose de particulier ? qu'est ce que c'est ?

```
head -n 69085 data.csv | tail -n 1
ou
sed -n 69085p data.csv
```

Douchanbe, capitale du Tadjikistan 0 :-) : Tout est à 0 ; votants comme inscrits, ce qui va créer des divisions par zero en cas de calculs de rapports

Q6. Quelle taille en octets et en nombre de lignes fait ce fichier ? (commande unix `wc`)

```
wc data.csv
   69243  215876 35127231 data.csv
```

69243 lignes, un peu plus de 35Mo de données

Q7. Supprimez de ce fichier la ligne d'entête ainsi que la ligne 69085, qui ne contiennent pas de données.

```
sed -i.old -e '1d' -e '69085d' data.csv
```

C'est important qu'ils sachent faire ces quelques requêtes unix de manipulation de données.

Certains vont essayer de supprimer les lignes avec un tableur, il y a toutes les chances que ça génère des erreurs, genre colonne en trop, séparateurs virgule, encodage etc ... **Il faut les obliger à passer par les commandes unix**

Q8. Afin de pouvoir manipuler facilement ces données, nous allons tout d'abord les importer dans notre SGBD. Créez une table de 98 colonnes avec le type de données approprié (un copier-coller vous sera surement utile ici)

```
CREATE TABLE import (
n0 text, n1 text, n2 text, n3 text, n4 text, n5 text, n6 text, n7 int, n8 int, n9 text, n10 int,
n11 text, n12 int, n13 text, n14 text, n15 int, n16 text, n17 text, n18 int, n19 text, n20 text,
n21 int, n22 text, n23 text, n24 text, n25 int, n26 text, n27 text,
n28 int, n29 text, n30 text, n31 text, n32 int, n33 text, n34 text,
n35 int, n36 text, n37 text, n38 text, n39 int, n40 text, n41 text,
n42 int, n43 text, n44 text, n45 text, n46 int, n47 text, n48 text,
n49 int, n50 text, n51 text, n52 text, n53 int, n54 text, n55 text,
n56 int, n57 text, n58 text, n59 text, n60 int, n61 text, n62 text,
n63 int, n64 text, n65 text, n66 text, n67 int, n68 text, n69 text,
n70 int, n71 text, n72 text, n73 text, n74 int, n75 text, n76 text,
n77 int, n78 text, n79 text, n80 text, n81 int, n82 text, n83 text,
n84 int, n85 text, n86 text, n87 text, n88 int, n89 text, n90 text,
n91 int, n92 text, n93 text, n94 text, n95 int, n96 text, n97 text);
```

Pour info
n0:dno; n1:Libelle; n2:circ; n3:libelle; n4:cno; n5:libelle; n6:bno; n7:Inscrits; n8:Abstentions; n9:Abs/Ins; n10:Votants; n11:Vot/Ins; n12:Blancs; n13:Blancs/Ins; n14:Blancs/Vot; n15:Nuls; n16:Nuls/Ins; n17:Nuls/Vot; n18:Exprimes; n19:Exp/Ins; n20:Exp/Vot; n21:cand; n22:Sexe; n23:Nom; n24:Prenom; n25:Voix; n26:Voix/Ins; n27:Voix/Exp ...puis multiples de 7 à partir de n21

Q9. Importez les données du fichier dans la table `import` précédemment créée à l'aide de la commande `\copy` de `psql` (`\help copy`)

```
\copy import from data.csv delimiter ';'
```

69241 lignes importées.

Si ils font un copier-coller, attention aux quotes !
Si on avait gardé l'entête, l'option CSV HEADER ; permettrait de la virer à l'importation, on aurait aussi pu garder l'encoage, réglable avec : ENCODING 'Latin1'

ON DOIT EN ETRE LA AU BOUT d'1/2H. sinon, donnez les solutions

Q10. Vérifiez à l'aide d'une requête SQL que vous avez le bon nombre de lignes (lignes initiales - 2)

```
SELECT COUNT(*) FROM import;
```

= 69241

Q11. Il y avait 11 candidats, que l'on retrouve sur la 1ère ligne affichée précédemment. Créez une table `candidats` avec les informations (nom,prénom,sexe) de chacun des candidats (là encore un copier-coller vous sera utile)

```
CREATE TABLE candidats(cand int,nom text,prenom text,sexe text);
INSERT INTO candidats VALUES(1,'DUPONT-AIGNAN','Nicolas','M');
INSERT INTO candidats VALUES(2,'LE PEN','Marine','F');
INSERT INTO candidats VALUES(3,'MACRON','Emmanuel','M');
INSERT INTO candidats VALUES(4,'HAMON','Benoît','M');
INSERT INTO candidats VALUES(5,'ARTHAUD','Nathalie','F');
INSERT INTO candidats VALUES(6,'POUTOU','Philippe','M');
INSERT INTO candidats VALUES(7,'CHEMINADE','Jacques','M');
INSERT INTO candidats VALUES(8,'LASSALLE','Jean','M');
INSERT INTO candidats VALUES(9,'MÉLENCHON','Jean-Luc','M');
INSERT INTO candidats VALUES(10,'ASSELINEAU','François','M');
INSERT INTO candidats VALUES(11,'FILLON','François','M');
```

Q12. De la même manière créez la table `communes(dno,cno,libelle)` que vous remplirez des données obtenues en `import`.

```
CREATE TABLE communes(dno,cno,libelle) AS SELECT DISTINCT n0,n4,n5 FROM import;
```

On a interêt à mettre des clés, sinon ça va ramer dans les requêtes !

Il devrait y avoir 35718 communes (c'est affiché à l'exec de la commande)

On note que
abstensions (col n8) = inscrits - votants = n7 - n10
exprimés (col n18) = Σ (suffrages des candidats)
blancs + nuls (n12 + n15) = votants - exprimes = n10 - n18

Q13. créez maintenant la table `bureaux(dno,cno,bno,inscrits,votants)` avec les données présentes dans `import`

On fait un peu chercher les numéros des bonnes colonnes ;-)
On note que la clé est dno,cno,bno

```
CREATE TABLE bureaux(dno,cno,bno,inscrits,votants)
       AS SELECT n0,n4,n6,n7,n10 FROM import;
```

Il y a 69241 lignes (c'est affiché à l'exec de la commande)

Q14. créez maintenant la table `votes` avec les données présentes dans `import`

```
CREATE TABLE votes (dno,cno,bno,cand,suffrages)
       AS SELECT n0,n4,n6,1,n25 FROM import
       UNION ALL SELECT n0,n4,n6,2,n32 FROM import
       UNION ALL SELECT n0,n4,n6,3,n39 FROM import
       UNION ALL SELECT n0,n4,n6,4,n46 FROM import
       UNION ALL SELECT n0,n4,n6,5,n53 FROM import
       UNION ALL SELECT n0,n4,n6,6,n60 FROM import
       UNION ALL SELECT n0,n4,n6,7,n67 FROM import
       UNION ALL SELECT n0,n4,n6,8,n74 FROM import
       UNION ALL SELECT n0,n4,n6,9,n81 FROM import
       UNION ALL SELECT n0,n4,n6,10,n88 FROM import
       UNION ALL SELECT n0,n4,n6,11,n95 FROM import;
```

Il devrait y avoir 761.651 résultats importés

Remarque: en toute rigueur, il faudrait faire en sorte que les id des candidats soient calculés automatiquement en fonction de leur nom, ce qui donnerait une union de 10 jointures de la forme:
`select n0,n4,n6,id,n25 from import i, candidat c where i.n17=c.nom`

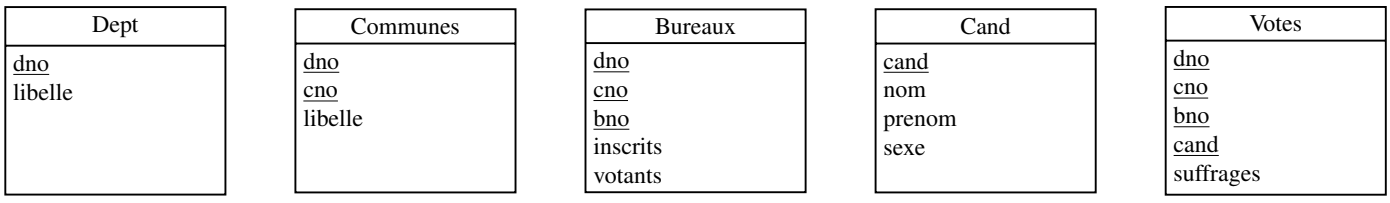
Q15. Créez la table `departements(dno,libelle)`. Vérifiez que vous avez bien le Nord et le Pas-de-Calais.

On leur laisse chercher les bonnes colonnes

```
CREATE TABLE departements(dno,libelle) AS SELECT DISTINCT n0,n1 FROM import;
```

107 lignes

On se retrouve donc maintenant avec les tables suivantes :



Exercice 2 : Optimisation

Q1. Afin de vérifier que tout est parfait, effectuez la jointure entre `departements`, `communes`, `bureaux`. Combien y-a t-il de lignes ? justifiez

```
SELECT COUNT(*) FROM (departements JOIN communes USING (dno)) JOIN bureaux USING(dno,cno) ;
```

69.241 autant que dans import.

Q2. Ajoutez à cette jointure la table votes. Combien y-a t-il de lignes ? justifiez

```
SELECT COUNT(*) FROM ((departements JOIN communes USING (dno)) JOIN bureaux USING(dno,cno)) JOIN votes USING (dno,cno,bno) ;
```

761.651 : Il y en a 11x plus que dans import. C'est comme dans `votes`. Dans import les candidats étaient tous sur une même ligne, dans cette jointure ils sont les uns en dessous des autres. Donc 11x plus de lignes.

Q3. Combien de temps met cette requête (on positionnera `\timing` pour cela.)

8 sec grosso-modo en TP

Q4. Vérifiez qu'il n'existe aucun index sur ces tables : `\di`

Q5. Ajoutez les clés primaires et étrangères à toutes les tables

```
ALTER TABLE departements ADD PRIMARY KEY(dno);
ALTER TABLE communes ADD PRIMARY KEY(dno,cno);
ALTER TABLE bureaux ADD PRIMARY KEY(dno,cno,bno);
ALTER TABLE candidats ADD PRIMARY KEY(cand);
ALTER TABLE votes ADD PRIMARY KEY(dno,cno,bno,cand);

ALTER TABLE communes ADD FOREIGN KEY (dno) REFERENCES departements(dno);
ALTER TABLE bureaux ADD FOREIGN KEY (dno,cno) REFERENCES communes(dno,cno);
ALTER TABLE votes ADD FOREIGN KEY (dno,cno,bno) REFERENCES bureaux(dno,cno,bno);
ALTER TABLE votes ADD FOREIGN KEY (cand) REFERENCES candidats(cand);
```

Q6. Combien de temps met la requete précédente ? Pourquoi ?

4 10è de sec

Q7. Regardez si des index ont été créés : `\di`

Exercice 3 : Un peu de fouille de données

Comparez les résultats obtenus avec la page Wikipedia correspondante.

Q1. Quel était le nombre total d'inscrits ?

```
SELECT SUM(inscrits) FROM bureaux;  : 47.582.183
```

Q2. Quel était le nombre total de votants ?

```
SELECT SUM(votants) FROM bureaux;  37.003.728
```

Q3. Calculez le taux de participation national (votants/inscrits)

```
SELECT SUM(votants)*100.0/SUM(inscrits) FROM bureaux;  77,768\%
```

Q4. Affichez les communes par longueur de nom décroissante. Quelle est la commune qui a le nom le plus long ?

```
SELECT * FROM communes ORDER BY LENGTH(libelle) DESC;
```

Saint-Remy-en-Bouzemont-Saint-Genest-et-Isson

Q5. Affichez les communes par longueur de nom croissante. Quelle est la commune qui a le nom le plus court ?

```
SELECT * FROM communes ORDER BY LENGTH(libelle) ASC;
```

Y

Q6. Existe-t-il des libellés de communes qui se répètent au moins 10 fois en France ? Lesquelles ?

```
SELECT libelle,COUNT(*) FROM communes GROUP BY libelle HAVING COUNT(*) >=10 ORDER BY COUNT(*);
```

5 lignes : Il y a 13 Sainte-Colombes

Q7. Est-ce qu'il y a 2 communes de même nom dans un même département ?

non. Il n'y en n'a pas

```
SELECT dno,libelle,COUNT(*) FROM communes GROUP BY dno,libelle HAVING COUNT(*)>1;
```

Q8. Affichez les départements par nombre de communes décroissant. Quel est le département avec le plus de communes ?

```
SELECT d.dno,d.libelle,COUNT(*) FROM communes JOIN departements AS d USING (dno)
GROUP BY d.dno,d.libelle ORDER BY COUNT(*) DESC;
```

pas de calais : 891

Q9. Affichez les départements par nombre de communes croissant. Quel est le département avec le moins de communes ?

```
idem avec ASC
```

Wallis et Futuna, Paris : 1

Q10. Affichez les communes par nombre d'inscrits croissant. Quelle est la commune avec le moins d'inscrits ?

```
SELECT cno,libelle, SUM(inscrits) FROM bureaux JOIN communes USING(dno,cno)
GROUP BY cno,libelle ORDER BY SUM(inscrits) ASC;
```

Attention à la jointure ! cno ne suffit pas !!
ou

```
SELECT n4,n5,SUM(n7) FROM import GROUP BY n4,n5  ORDER BY SUM(n7)  ASC;
```

Douaumont, Leménil-Mitry : 6

Q11. Affichez les communes par nombre d'inscrits décroissant. Quelle est la commune avec le plus d'inscrits ?

idem avec DESC paris, Marseille

Q12. Afficher les noms des communes avec leur nombre de bureaux et le nombre total d'inscrits, par order décroissant du nombre de bureaux

```
SELECT cno, libelle, COUNT(*), SUM(inscrits)
FROM bureaux JOIN communes USING (dno,cno)
GROUP BY cno, libelle
ORDER BY COUNT(*) DESC;t
```

Attention au critère de jointure !! si on ne prend que cno le calcul explose !
Paris : 896 bureaux

Q13. En ne prenant en compte que les communes où il y a plusieurs bureaux, afficher par commune le nombre moyen d'inscrits par bureau, trié par nombre décroissant

```
SELECT cno, libelle, COUNT(*), SUM(inscrits), SUM(inscrits)/COUNT(*)
FROM bureaux JOIN communes USING (dno,cno)
GROUP BY cno, libelle
HAVING COUNT(*) >1
ORDER BY SUM(inscrits)/COUNT(*) DESC;
```

Grosso-modo la même que la précédente.

Exercice 4 : Analyse des votes

Q1. Afin de faciliter l'étude des résultats par commune, créez une vue `vuecommunes(dno,cno,libelle,inscrits,votants)` permettant de voir les données des votes de manière agrégée par commune.

```
CREATE VIEW vuecommunes AS
SELECT dno,cno,libelle,SUM(inscrits) AS inscrits, SUM(votants) AS votants
FROM bureaux JOIN communes USING(dno,cno)
GROUP BY dno,cno,libelle;
```

Q2. Vérifiez que le nombre de lignes de cette vue est bien identique au nombre de communes

```
SELECT COUNT(*) FROM communes;
SELECT COUNT(*) FROM vuecommunes;
```

Autant que de communes 35.718

Q3. Affichez les communes "hors de france" par nombre d'inscrits décroissant. Quelle est celle avec le plus d'inscrits ?

Hors de france : code ZZ dans departements

```
SELECT * FROM vuecommunes WHERE dno='ZZ'  ORDER BY inscrits DESC;
```

Geneve
c'est super interessant de regarder la liste ;-)

Q4. Affichez uniquement la ou les communes qui sont au taux de participation (votants/inscrits) le plus faible, avec le taux de participation correspondant.

```
SELECT *, votants*1.0 / inscrits
FROM vuecommunes
WHERE (votants*1.0 / inscrits) = (SELECT MIN(votants*1.0/inscrits) FROM vuecommunes);
```

1 commune : Saana : 0%
Si on ne vire pas 'Douchanbe' on a une division par zero

Q5. Affichez uniquement la ou les communes qui sont au taux de participation (votants/inscrits) le plus élevé

Il y en a 45 : dont Tannières

Q6. Quel est ce taux de participation maximum ?

100%

Q7. A des fins de statistiques et de comparaison, on cherche 2 communes assez conséquentes, avec plus de 2000 inscrits et plus de 2000 votants, et qui auraient exactement le même nombre de votants et d'inscrits (donc parfaitement simulaires)

```
SELECT * FROM vuecommune c1 JOIN vuecommune c2 USING(votants,inscrits)
WHERE c1.libelle <> c2.libelle AND c1.inscrits>2000 AND c1.votants>2000 AND c1.cno > c2.cno;
```

ça se produit 3 fois ! (sans symétrique)

Q8. Combien Benoit Hamon a t-il eu de voix au total ?

```
SELECT SUM(suffrages)
FROM votes
WHERE cand = (SELECT cand FROM candidats WHERE nom='HAMON');
```

2.291.288

Q9. Donnez le classement national des candidats avec leur nom, ordonné par total des suffrages exprimés décroissant.

```
SELECT nom,SUM(suffrages)
FROM votes AS v JOIN candidats AS C USING (cand)
GROUP BY nom
ORDER BY SUM(suffrages) DESC;
```

Q10. Dans quelle commune CHEMINADE a eu le plus de voies ?

```
SELECT * FROM (votes JOIN candidats USING (cand))
                     JOIN communes USING (dno,cno)
WHERE nom='CHEMINADE' ORDER BY suffrages DESC;
```

154 voies à Geneve

Q11. Donnez le cumul des voix décroissant par candidat et uniquement pour le département du nord

```
SELECT nom, SUM(suffrages)
FROM votes AS v , candidats AS C
WHERE C.cand=v.cand AND dno LIKE '59%'
GROUP BY nom
ORDER BY SUM(suffrages) DESC;
```

LE PEN, MELANCHON, MACRON

Q12. Affichez les résultats des candidats pour la commune de Villeneuve d'Ascq

```
SELECT nom, libelle, SUM(suffrages)
FROM votes AS v , candidats AS C , communes AS co
WHERE C.cand=v.cand AND v.dno=co.dno
AND v.cno=co.cno AND libelle LIKE 'Villeneuve-d''Ascq'
GROUP BY nom,libelle;
```

Mélanchon en tête

Q13. Et là où vous habitez, quel est l'ordre des candidats ?

Exercice 5 : Exporter les données

Q1. Exportez la tables `candidats` dans un fichier `candidats.csv` (`\help copy`)

```
\COPY candidats TO 'candidats.csv' DELIMITER';'
\COPY communes TO 'communes.csv' DELIMITER';'
\COPY votes TO 'votes.csv' DELIMITER';'
\COPY bureaux TO 'bureaux.csv' DELIMITER';'
\COPY departements TO 'departements.csv' DELIMITER';'
```

Q2. Exportez la tables `communes` dans un fichier `communes.csv`

Q3. De manière identique exporter les tables `votes`, `bureaux`, `departements` dans leurs fichiers csv respectifs.

Q4. Quelle somme en octets fait maintenant la somme de ces fichiers ? Comparez la avec ta taille du fichier initial

```
wc -c {communes,candidats,votes,bureaux,departements}.csv
```

On passe de 35M à 15M , donc un gain de 43%

Q5. Supprimez les tables et vues de ce TP.