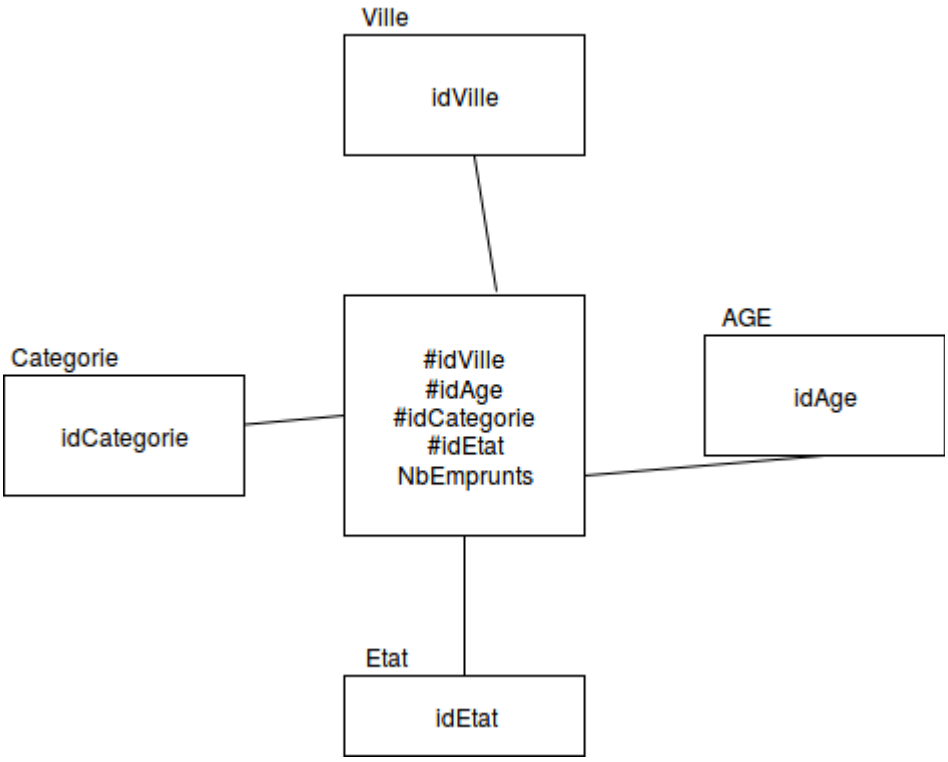
Clément LOUBIERE
Tristan RIVIERE

1.



2.

a) Select a.ville, a.age, lv.categorie, ex.etat, Count(*) as NbEmprunts
FROM abonne a, livre lv, emprunt e, exemplaire ex
WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
GROUP BY a.ville, a.age, lv.categorie, ex.etat

b)

Select a.ville, a.age, lv.categorie, ex.etat, Count(*) as NbEmprunts
FROM abonne a, livre lv, emprunt e, exemplaire ex
WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
GROUP BY ROLLUP(a.ville, a.age, lv.categorie, ex.etat)

Affiche toutes les combinaisons existante en faisant à chaque fois varier le paramètre le plus à droite.

c)
```
Select a.ville, a.age, lv.categorie, ex.etat, Count(*) as NbEmprunts
        FROM abonne a, livre lv, emprunt e, exemplaire ex
        WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
        GROUP BY CUBE( a.ville, a.age, lv.categorie, ex.etat)
```
Génère le cube a n dimension (n etant le nombres de varaibles passées dans la clause group by cube).
Génère toutes les combinaisons possible (meme si inexistante) selon l'ensemble des dimensions passées en paramètre.

d)
```
Select a.ville, a.age, lv.categorie, ex.etat, Count(*) as NbEmprunts
        FROM abonne a, livre lv, emprunt e, exemplaire ex
        WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
        GROUP BY GROUPINGS SETS( a.ville, a.age, lv.categorie, ex.etat)
```
Calcule ttous les sous cubes possibles.

e)
On remplace pour chaque variable du select par DECODE(valeur, null, 'toutes valeur confondues', valeur). Ainsi on dis "pour la valeur, si elle est 'null' alors l'afficher comme 'Toutes valeurs confondues' sinon afficher sa valeur). Attention Decode ne fais pas la distinction entre un null retourné par une clause (signifiant bien 'toutes valeurs confondues') d'un null entré en brute dans l'une des tables.

1. Densité du cube :
Densité = Nombre de requetes retourné par le group by / (Nombre ville x Nombres d'ages x Nombre de catégories x Nombre d'état)
Densité = 17 / (3 x 8 x 3 x 4)
Densité = 4%

2.a)
```
Select a.ville, a.age, lv.categorie, ex.etat, COUNT(*) as NbEmprunts, RANK()
OVER(PARTITION BY a.ville) as RANK
        FROM abonne a, livre lv, exemplaire ex, emprunt e
        WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
        GROUP BY CUBE( a.ville, a.age, lv.categorie, ex.etat)
```
On retourne les villes ayant le plus d'emprunt en premier.

b)
```
Select a.ville, a.age, lv.categorie, ex.etat, COUNT(*) as NbEmprunts,
RATIO_TO_REPORT(COUNT(*)) OVER(PARTITION BY a.ville) as
PourcentageDempruntParVille
        FROM abonne a, livre lv, exemplaire ex, emprunt e
        WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
        GROUP BY CUBE( a.ville, a.age, lv.categorie, ex.etat)
```
c) Pas a faire
d)
```
SELECT sr.categorie, max(sr.NbEmprunt)
        From ( SELECT a.ville, a.age, lv.categorie, ex.etat, COUNT(*) as NbEmprunts
                FROM abonne a, livre lv, empreunt e, exemplaire ex
                WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
                GROUP BY a.ville, a.age, lv.categorie, ex.etat
                ) sr
        GROUP BY sr.categorie
```

3.a) SELECT *
FROM (Select a.ville, a.age, lv.categorie, ex.etat, Count(*) as NbEmprunts, RANK() OVER(ORDER BY COUNT(*) DESC) as rank
FROM abonne a, livre lv, emprunt e, exemplaire ex
WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
GROUP BY (a.ville, a.age, lv.categorie, ex.etat)) sr
WHERE sr.rank <=2

b)SELECT *
FROM (Select a.ville, a.age, lv.categorie, ex.etat, Count(*) as NbEmprunts, RANK() OVER(ORDER BY COUNT(*) ASC) as rank
FROM abonne a, livre lv, emprunt e, exemplaire ex
WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
GROUP BY (a.ville, a.age, lv.categorie, ex.etat)) sr
WHERE sr.rank <=2 AND rownum <=2

c) SELECT *
FROM (Select lv.categorie, Count(*) as NbEmprunts, RANK() OVER(ORDER BY COUNT(*) DESC) as rank
FROM abonne a, livre lv, emprunt e, exemplaire ex
WHERE a.num_ab= e.num_ab AND e.num_ex = ex.numero AND ex.isbn = lv.isbn
GROUP BY (a.ville, a.age, lv.categorie, ex.etat)) sr
WHERE sr.rank <=2

4)a) Les vues matérialisées servent à faire une duplication des données d'une requete. Cela permet de stocker les résultats d'une requete particulièrement longue et/ou complexe.

b)
CREATE MATERIALIZED VIEW MV
AS SELECT * FROM TABLE;

d)refresh on demand
start with sysdate next sysdate + 1
AS SELECT * FROM TABLE

f) non la vue n'est pas à jour

h) oui les modifications ont été prises en compte

5)a)C'est le fait qu'une valeur puisse changer :
on peut changer de numéro de téléphone, d'adresse, de travail,...

b) L'état d'un exemplaire peut se dégrader

c) On peut choisir d'écraser l'ancienne valeur.
Problème : on ne sait pas quand la donnée a changé de valeur ou si elle a changé.

Versionner les changements de valeurs
Problème : prend plus de place dans la base

6)a)Un ETL récupère des données venant de plusieurs endroit, les transforme puis les stocke dans une base de données.

b)Ils peuvent normaliser les données en changeant les valeurs, unités, normes, ...

c) Talend, SynchroDB, ...