Bases de Données
Les unités de valeurs

Vous disposez du schéma relationnel suivant :

Les schémas des relations :

`ENSEIGNANTS ( N°Ens , NomEns , Grade , DateNaissance )`
`UVS ( N°UV , NomUV , NombreHeuresUV , N°Ens )`
`RÉSULTATS ( N°Étu , N°UV , Note )`
`ÉTUDIANTS ( N°Étu , NomÉtu , Adresse , CodePostal , Ville , DateNaissance )`

Les contraintes de clé (primaire) :

L'attribut `ENSEIGNANTS.N°Ens` est la clé de `ENSEIGNANTS`
L'attribut `UVS.N°UV` est la clé de `UVS`
Le couple d'attributs `( RÉSULTATS.N°Étu , RÉSULTATS.N°UV )` est la clé de `RÉSULTATS`
L'attribut `ÉTUDIANTS.N°Étu` est la clé de `ÉTUDIANTS`

Les contraintes d'intégrité référentielles :

L'attribut `UVS.N°Ens` référence l'attribut `ENSEIGNANTS.N°Ens`
… et donc `valeurs(UVS.N°Ens)` $\subseteq$ `valeurs(ENSEIGNANTS.N°Ens)`
L'attribut `RÉSULTATS.N°Étu` référence l'attribut `ÉTUDIANTS.N°Étu`
… et donc `valeurs(RÉSULTATS.N°Étu)` $\subseteq$ `valeurs(ÉTUDIANTS.N°Étu)`
L'attribut `RÉSULTATS.N°UV` référence l'attribut `UVS.N°UV`
… et donc `valeurs(RÉSULTATS.N°UV)` $\subseteq$ `valeurs(UVS.N°UV)`

Les contraintes d'unicité :

L'attribut `UVS.NomUV` est unique (sans répétition)

Les contraintes existentielles :

Seuls les attributs `ÉTUDIANTS.Ville` et `RÉSULTATS.Note` sont facultatifs i.e. qu'ils peuvent être indéterminés

Écrivez en SQL (et parfois en calcul relationnel et en algèbre relationnelle) les requêtes permettant de répondre aux demandes suivantes.

Requêtes de base (sélection, projection, tri, jointure) **en SQL, en calcul relationnel et en algèbre relationnelle**.

1. *Toutes les informations sur tous les étudiants.*
2. *Les noms des étudiants (sans répétitions).*
3. *Les noms des étudiants.*
4. *Les étudiants bordelais.*
5. *Les numéros et noms des étudiants dont le nom contient la lettre Y ou qui sont nés entre le 1er janvier 1990 et le 31 décembre 1999.*
6. *Toutes les informations sur les étudiants et les unités de valeurs qu'ils ont suivies.*
7. *Les notes de l'étudiant numéro 4239 triées en ordre décroissant.*
8. *Les numéros d'étudiants, numéros des unités de valeurs et notes de l'étudiant DUPOND.*
9. *Les enseignants (sans répétitions) de l'étudiant numéro 4239.*
10. *Les enseignants de l'étudiant DUPOND triés en ordre alphabétique sur le nom.*
11. *Les étudiants de l'enseignant numéro 87.*
12. *Les numéros et noms des étudiants habitant une ville dont la valeur est renseignée (i.e. le champ Ville a une valeur et non la pseudo-valeur d'indétermination).*
13. *Les enseignants intervenant dans des unités de valeurs (au moins une fois).*

Requêtes sur les regroupements.

14. *Le nombre d'enseignants.*
15. *Pour chaque étudiant, le nombre de ses notes.*
16. *Pour chaque étudiant, sa note minimale, sa note maximale, l'écart entre ses deux notes extrêmes et sa moyenne.*
17. *Les enseignants intervenant dans exactement deux unités de valeurs.*
18. *Les enseignants intervenant dans au moins trois unités de valeurs, en affichant le numéro et le nom des enseignants ainsi que le nombre d'unités de valeurs ; le résultat est trié en ordre décroissant du nombre d'unités de valeurs et en ordre croissant sur le nom.*
19. *Les étudiants ayant la moyenne générale.*
20. *Les étudiants bordelais dont le volume horaire est inférieur à 500 heures (toutes unités de valeurs confondues).*

Requêtes sur les opérations ensemblistes **en SQL, en calcul relationnel et en algèbre relationnelle**.

21. *Les noms des étudiants et les noms des enseignants (sans répétitions, sur une seule colonne).*
22. *Les noms communs aux étudiants et aux enseignants (sans répétitions).*
23. *Les noms des étudiants qui ne sont pas des noms d'enseignants (sans répétitions).*
24. *Les numéros et noms des étudiants bordelais ou gradignanais.*
25. *Les enseignants n'intervenant dans aucune des unités de valeurs.*

Requêtes imbriquées (i.e. sous-requêtes).

26. *Les noms communs aux étudiants et aux enseignants* (trouver une solution différente de celle trouvée précédemment).
27. *Les enseignants ayant le grade le plus élevé.*
28. *Les étudiants qui habitent la(les) ville(s) où résident le plus d'étudiants.*
    Par exemple, si pour cinq étudiants, deux sont bordelais, deux sont gradignanais et un est talençais, la réponse correspond aux quatre étudiants de Bordeaux ou de Gradignan.
29. *Les étudiants n'ayant pas l'enseignant numéro 87.*
30. *Les enseignants n'intervenant dans aucune des unités de valeurs* (trouver une solution différente de celle trouvée précédemment).

Requêtes utilisant des instructions SQL récentes.

31. *Que fait la requête suivante selon que* ◁clause de regroupement « classique » ou *OLAP*▷ *vaut* N°Étu , N°Ens *ou* rollup(N°Étu , N°Ens) *ou* cube(N°Étu , N°Ens) *ou* grouping sets (N°Étu , N°Ens , ()) *?*

```
select N°Étu , N°Ens , count(*) , avg(Note)
from ÉTUDIANTS
natural join RÉSULTATS
natural join UVS
natural join ENSEIGNANTS
group by ◁clause de regroupement « classique » ou OLAP▷
order by N°Étu null last , N°Ens null last
```

32. *Que fait la requête ?*

```
select rank() over(order by Note desc) , Note , É.*
from ÉTUDIANTS É
natural join RÉSULTATS
natural join UVS
where NomUV = 'Bases de données'
order by 1 , N°Étu
```

33. *Que fait la requête ?*

```
select Grade , N°Ens , NomEns , rank() over(partition by Grade order by N°Ens) ,
       count(*) over(partition by Grade) , count(*) over()
from ENSEIGNANTS
order by Grade , N°Ens
```

34. *Que fait la requête ?*

```
select É.* , case when Note >= 16 then 'Très Bien'
                  when Note >= 14 then 'Bien'
                  when Note >= 12 then 'Assez Bien'
                  else                 null
             end
from ÉTUDIANTS É
natural join RÉSULTATS
natural join UVS
where NomUV = 'Bases de données'
```

35. *Que fait la requête ?*

```
select extract(year from DateNaissance)
from ENSEIGNANTS
intersect
select extract(year from DateNaissance)
from ÉTUDIANTS
```

36. *Que fait la requête ?*

```
select N°Ens , NomEns , nullif(Grade,0) , coalesce(DateNaissance,'inconnue')
from ENSEIGNANTS
where NomEns similar to '[C-F]||[AEIOUY]%[DT]||[X-Z]'
```

37. *Que fait la requête ?*

```
with ÉTUDIANTS_PAR_VILLE ( Ville , NbÉtuParVille ) as ( select Ville , count(*)
                                                        from ÉTUDIANTS
                                                        group by Ville          )
select *
from ÉTUDIANTS
where Ville in ( select Ville
                 from ÉTUDIANTS_PAR_VILLE
                 where NbÉtuParVille = ( select max(NbÉtuParVille)
                                         from ÉTUDIANTS_PAR_VILLE  ) )
```

Requêtes avancées.

38. *Indiquez ce que fait la requête ci-dessous et écrivez (sans* `match`*) une requête équivalente :*

```
select *
from ÉTUDIANTS
where ( NomÉtu , DateNaissance ) match ( select NomEns , DateNaissance
                                         from ENSEIGNANTS              )
```

39. *Indiquez ce que fait la requête ci-dessous et écrivez (à l'aide d'opérations ensemblistes) une requête équivalente :*

```
select N°Étu , N°UV , count(*) , avg(Note)
from RÉSULTATS
group by rollup( N°Étu , N°UV )
```

40. *Que fait la requête ?*

```
select É.* , count(N°UV) , sum(Note)-count(Note)*avg(Note)
from ÉTUDIANTS É
left outer natural join RÉSULTATS
group by É.*
```

41. *Que fait la requête ?*

```
select distinct NomEns
from ENSEIGNANTS , ÉTUDIANTS
where NomEns = NomÉtu
```

42. *Que fait la requête ?*

```
select *
from ÉTUDIANTS
where exists ( select 33
               from RÉSULTATS
               where RÉSULTATS.N°Étu = ÉTUDIANTS.N°Étu and Note = 20 )
```

43. *Que fait la requête ?*

```
select count(*) , UV.*
from UVS UV , UVS COPIE_UV
where UV.NomUV >= COPIE_UV.NomUV
group by UV.*
```

44. *Que fait la requête ?*

```
select *
from ENSEIGNANTS
where ( select count(*)
        from UVS
        where UVS.N°Ens = ENSEIGNANTS.N°Ens ) between 3 and 6
```

45. *Que fait la requête ?*

```
select distinct E.*
from ENSEIGNANTS E , ENSEIGNANTS COPIE_E
where ( E.DateNaissance , E.DateNaissance + interval '5' YEAR ) overlaps
      ( COPIE_E.DateNaissance - interval '5' YEAR , COPIE_E.DateNaissance )
```

46. *Que fait la requête ?*

```
select *
from ENSEIGNANTS E1
where not exists ( select 'Azerty'
                   from ENSEIGNANTS E2
                   where E2.Grade > E1.Grade )
```

47. *Que fait la requête ?*

```
select É.N°Étu , É.NomÉtu
from ÉTUDIANTS É , ÉTUDIANTS COPIE_É
where É.NomÉtu = COPIE_É.NomÉtu
group by É.*
having count(*) > 1
```

48. *Que fait la requête*

```
select *
from UVS
where unique ( select ◁projection sous-requête▷
               from RÉSULTATS
               where RÉSULTATS.N°UV = UVS.N°UV and Note is not null )
```

selon que `◁projection sous-requête▷` *vaut* `N°Étu` *ou* `N°UV` , `N°Étu` *ou* `N°UV` *ou* `null` *ou* `true` *?*

49. *Les enseignants n'intervenant dans aucune des unités de valeurs* (trouver une solution différente de celles trouvées précédemment).
50. *Tous les enseignants, avec le nombre d'unités de valeurs (éventuellement zéro) dans lesquelles ils interviennent.*

51. *Que fait la requête ?*

```
select *
from ENSEIGNANTS
where N°Ens <> any ( select N°Ens
                     from UVS     )
```

52. *Que fait la requête ?*

```
select *
from ÉTUDIANTS
where NomÉtu <= all ( select NomÉtu
                      from ÉTUDIANTS
                      natural join RÉSULTATS
                      where Note = 20        )
```

Requêtes de mise à jour.

53. *Insérer un enseignant.*
54. *Modifier le grade de tous les enseignants en leur ajoutant 5 points.*
55. *Supprimer tous les étudiants bordelais.*
56. *Affecter à zéro la note de tous les étudiants n'ayant pas participé à l'unité de valeur de Bases de Données (i.e. insérer uniquement les notes n'existant pas, sans modifier celles existant déjà, que la valeur de la note soit renseignée ou non).*
    Par exemple, pour les quatre (seuls) étudiants de numéros 1, 2, 3 *et* 4, *et en supposant que* $\pi_{N°Étu,Note}$ (RÉSULTATS ⋈ $\sigma_{NomUV='Bases\ de\ données'}$ (UVS) = { (1,20) , (2,0) , (3,non renseigné) }, *il faut alors insérer* (4,$\pi_{N°UV}$($\sigma_{NomUV='Bases\ de\ données'}$(UVS)),0) *dans* RÉSULTATS.
57. *En une seule commande SQL, pour tous les étudiants, insérer parmi les enseignants tous ceux dont le numéro d'étudiant n'est pas déjà un numéro d'enseignant (les numéro, nom et date de naissance sont repris tandis que le grade est mis à zéro) et sinon modifier le grade des enseignants les moins gradés en l'augmentant d'un point*.