TP4

Maël PEROT 22113572t

Exercice 1 :

1)

a)

```
SELECT S.nom AS 'nom de station', R.ventMax, R.dateRel AS date FROM station_meteo S
INNER JOIN releve R ON R.idStation=S.idStation
WHERE R.VentMax=(SELECT MAX(ventMax) FROM releve)
```

b)

```
SELECT P.nom, P.prenom FROM personne P
WHERE P.idStation IN (SELECT idStation FROM personne
                      GROUP BY idStation
                      HAVING COUNT(*)=1)
```

2)

a)

```
SELECT S.nom AS Station, MONTH(R.dateRel) AS mois, YEAR(R.dateRel) AS annee,
SUM(R.PrecipJour) AS ‘Cumul des precipitations’  FROM station_meteo S
INNER JOIN releve R ON R.idStation=S.idStation
GROUP BY Station, mois, annee
ORDER BY annee, mois
```

b)

```
SELECT ROUND(AVG(R.Tmax), 1) AS 'moyenne des Tmax', YEAR(R.dateRel) AS annee, S.nom
AS station FROM releve R
INNER JOIN station_meteo S ON S.idStation=R.idStation
WHERE MONTH(R.dateRel)=8
GROUP BY annee
```

3)

a)

Ces mots clés permettent de créer une vue en lui donnant un nom. Cette vue s’appelle donc moyennesMois. Une vue est une requête qui existe en même temps que la table qui l’a créée.
Une vue sert a limiter les données visibles et donne l’opportunité de sélectionner lesquelles.

b)

```
SELECT moyTmin, moyTmax, moyVentMax FROM `moyennesMois`
WHERE mois=12 AND annee=1999 AND station LIKE 'Haguenau Météo'
```

c)

Cette requête devrait insérer les valeurs dans la table moyennesMois mais on récupère une erreur car on ne peut rien insérer dans une vue.

d)

```
DROP VIEW ‘moyennesMois’
```

Exercice 2 :

1)

On ne peut pas supprimer les personnes car elles sont liees dans une autre table. Il faut garder une coherence.

2)

```
DELETE FROM personne
WHERE idPersonne NOT IN (SELECT DISTINCT(idPersonne) FROM releve)
```

3)

```
INSERT INTO station_meteo VALUES (null ,'Meteo Marseille', 'COM13055', 5)
```

4)
```
UPDATE station_meteo SET altitude=6 WHERE idStation=16
```
5)
```
ALTER TABLE personne ADD email varchar(255)
```
6)
Pour un SGBD qui gère les contraintes CHECK la première adresse mail serait refusée car elle ne contient pas de @, la deuxième serait en revanche acceptée.

Exercice 3 :

1)
La requête générée est de type CREATE TABLE.
2)
```
ALTER TABLE image
ADD FOREIGN KEY (idPersonne, idReleve) REFERENCES personne(idPersonne),
releve(idReleve);
```
5)
```
SELECT dateImg, P.nom AS 'image deposee par' FROM image
NATURAL JOIN personne P
```
6)
```
ALTER TABLE `releve`
  DROP `Tmoy`;
```
7)

a)

Le contenu de la table n'a pas changé. Il changera seulement si on fait un commit pour effectuer les transactions demandées.

b)

Le contenu de la table a changé.

c)

Dans la table personne on a bien le nombre de relevés effectués par la personne avec l'id 1 donc toute la transaction s'est effectuée.

d)

Il ne se passe rien

Exercice 4 :

1)
```
SELECT S.nom, YEAR(dateRel) AS annee, COUNT(*) AS 'nb jours de pluie' FROM releve
INNER JOIN station_meteo S ON S.idStation=releve.idStation
WHERE PrecipJour>5
GROUP BY S.idStation, annee
```
2)
```
SELECT S.nom, YEAR(dateRel) AS annee, COUNT(*) AS 'nbJoursDePluie' FROM releve
INNER JOIN station_meteo S ON S.idStation=releve.idStation
WHERE PrecipJour>5
GROUP BY S.idStation, annee
HAVING nbJoursDePluie<10
```
3)
```
SELECT C.nom AS ville, MONTH(dateRel) AS mois, YEAR(dateRel) AS annee,
SUM(precipJour) AS cumul FROM releve R
INNER JOIN station_meteo S ON S.idStation=R.idStation
INNER JOIN commune C ON C.numInsee=S.numInsee
GROUP BY C.nom, mois, annee
HAVING cumul>1000
```