INFO-H-303 Bases de données
Séance d'exercices 8
SQL (2)

22 mars 2018

Quantificateur universel

Formes équivalentes :

- Les employés qui travaillent sur tous les projets.
- Les employés pour lesquels il n'existe pas de projet sur lequel ils ne travaillent pas.

Traduction en SQL :

```
SELECT e.FName, e.LName
FROM Employee e
WHERE NOT EXISTS
   ( SELECT *
     FROM Project p
     WHERE NOT EXISTS
        ( SELECT *
          FROM WorksOn w
          WHERE w.ESSN = e.SSN AND w.PNo = p.PNumber) )
```

SQL : comparaison avec une sous-requête

- `<attribut> <comparateur>(<sous-requête>)`
- Exemple :

```
SELECT p.NP
FROM P p
WHERE p.Poids <= ( SELECT MIN(p2.Poids)
                   FROM P p2 )
```

- Avec certains DBMS, tels que SQLite, la sous-requête peut renvoyer plusierus entrées
- Sinon, avec MySQL `ANY` permet de vérifier si au moins une valeur de la liste satisfait la condition

```
SELECT p.NP
FROM P p
WHERE NOT (p.Prix >= ANY (SELECT p2.Poids
                          FROM P p2))
```

SQL : agrégation

- Les fonctions `COUNT()`, `SUM()`, `MIN()`, `MAX()`, `AVG()` permettent d'agréger les données.
- Résultat : une ligne par partition (ou par relation si pas de partition).
- Exemple : moyenne des salaires

```
SELECT AVG(Salary) FROM Employee
```

- Exemple : produit le plus légér :

```
SELECT p.NP FROM P p
WHERE p.Poids = ( SELECT MIN(p2.Poids) FROM P p2 )
```

- La requête suivante est illégale : pourquoi ?

```
SELECT SSN, MAX(Salary) FROM Employee
```

SQL : group by

- La clause `GROUP BY <attributs>` permet de partitionner une relation.
- Il y aura une partition par combinaison de valeurs des attributs.

```
SELECT A1, ..., An, AGG(An+1), ..., AGG(Am)
FROM ... WHERE ...
GROUP BY A1, ..., An
HAVING condition
```

- Les attributs du `SELECT (A1, ..., An)` doivent être des clés de groupement.
- La condition du `HAVING` porte sur les partitions, donc sur les attributs `A1, ..., An` et sur des agrégations.

SQL : group by

```
SELECT A1, ..., An, AGG(An+1), ..., AGG(Am)
FROM ... WHERE ...
GROUP BY A1, ..., An
HAVING condition
```

Ordre intuitif d'évaluation :

1. Evaluation du `FROM... WHERE ...`
2. Partitionnement selon les attributs du `GROUP BY`
3. Application de la condition du `HAVING`
4. Evaluation du `SELECT`

SQL : group by

Moyenne des salaires par département

```
SELECT DNo, AVG(Salary)
FROM Employee
GROUP BY DNo
```

Moyenne des salaires pour les départements de moins de 3 employés

```
SELECT DNo, AVG(Salary)
FROM Employee
GROUP BY DNo
HAVING count(*) < 3
```

SQL : exercices

- 12, 17, 18, 19, 25, 26
- Donner le numéro des fournisseurs qui ont vendu plus de 100 produits.
- Pour chaque fournisseur de Londres qui vend au moins 3 produits différents, donner le numéro du fournisseur et la quantité de produits vendus.
- 21, 22, 23, 24 (insertions ...)

SQL : exercices

Donner le nombre d'usines approvisionnées par le fournisseur 1

```
SELECT COUNT(distinct l.NU)
FROM PUF l
WHERE l.NF = 1
```

SQL : exercices

Pour chaque produit livré à une usine, donner le numéro du produit, celui de l'usine et la quantité totale livrée

```
SELECT l.NP, l.NU, SUM(l.Quantite)
FROM PUF l
GROUP BY l.NP, l.NU
```

SQL : exercices

Donner le numéro des fournisseurs qui ont vendu plus de 100 produits

```
SELECT l.NF
FROM PUF l
GROUP BY l.NF
HAVING SUM(l.Quantite) > 100
```

SQL : exercices

Pour chaque fournisseur de Londres qui vend au moins 3 produits différents, donner le numéro du fournisseur et la quantité de produits vendus

```
SELECT l.NF, SUM(l.Quantité)
FROM PUF l, F f
WHERE l.NF = f.NF and f.VilleF='Londres'
GROUP BY l.NF
HAVING COUNT(distinct l.NP) >= 3
```