Opération DIVISION

Huit opérations de base de l'algèbre relationnelle

PROJECTION	SELECT [ALL] [DISTINCT] liste d'attributs FROM table
SELECTION	SELECT liste d'attributs FROM table WHERE condition
JOINTURE	SELECT liste d'attributs FROM table1 **JOIN** table2 ON table1.attribut1=table2.attribut1 ...*ou* WHERE
DIVISION	???
UNION	SELECT liste d'attributs FROM table **UNION** SELECT liste d'attributs FROM table
INTERSECTION	SELECT liste d'attributs FROM table **INTERSECT** SELECT liste d'attributs FROM table *ou* WHERE avec NOT IN

DIFFERENCE	SELECT liste d'attributs FROM table **EXCEPT** SELECT liste d'attributs FROM table
PRODUIT	SELECT * FROM table1, table2 (pas de where) *ou* SELECT * FROM table1 **CROSS JOIN** table2

Il n'existe pas en SQL d'équivalent direct à la division !

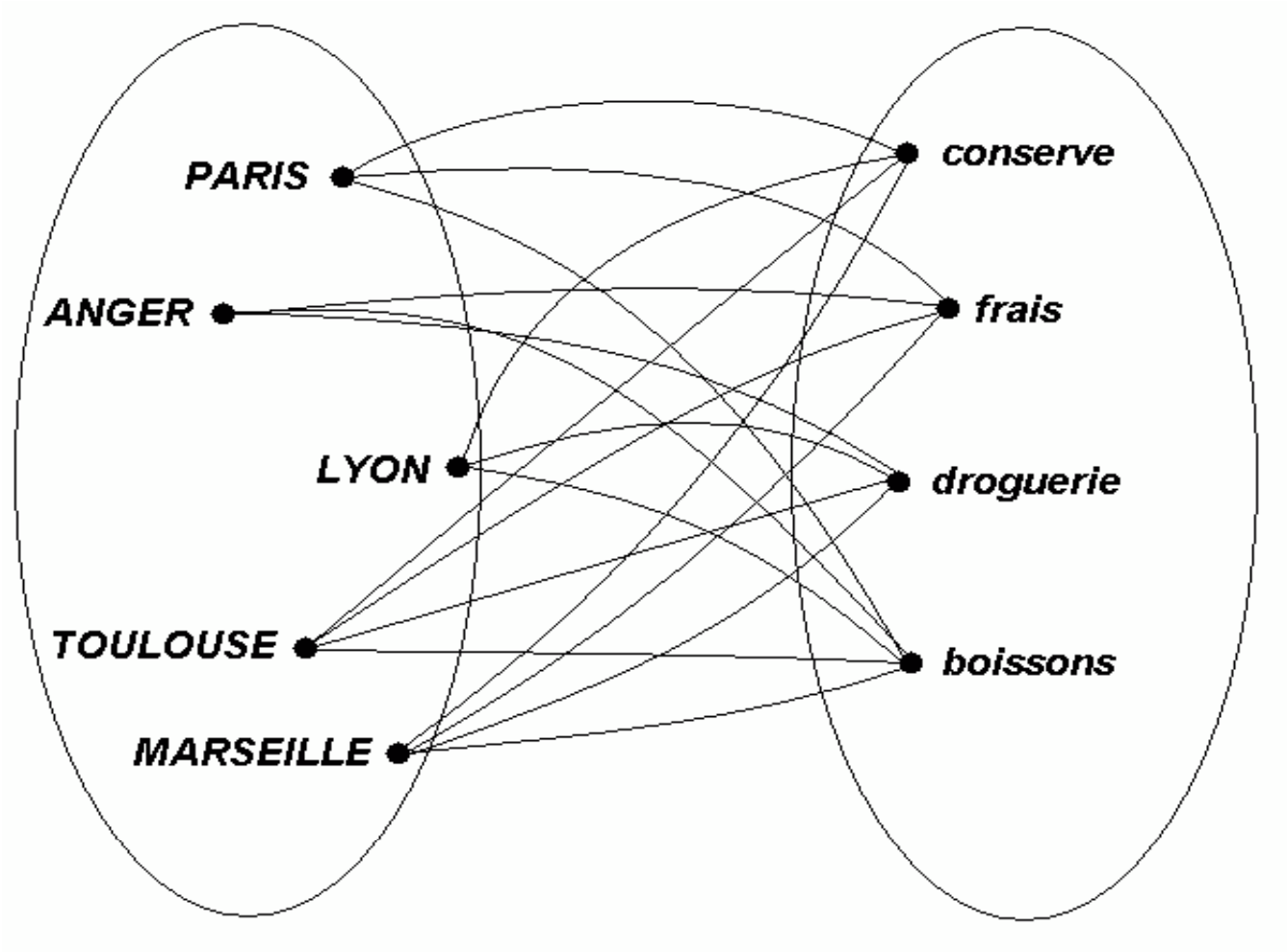
→ Recherche d'une autre solution

Exemple 1 :

Une entreprise de la grande distribution possède des entrepôts dans différentes villes de France.
Ces entrepôts peuvent abriter les produits de différents rayons.

Question: ***quels sont les entrepôts capables de servir TOUS les rayons ?***

Exemple 1 (suite) :



Exemple 1 (suite) :

Réponse : TOULOUSE et MARSEILLE

Il suffit de trouver les entrepôts qui sont reliés à TOUS les rayons.
Table des rayons :
- nom : T_RAYON
- attribut RYN

Tables des entrepôts :
- nom : T_ENTREPOT
- attributs :
  - VILLE_ETP
  - RAYON_RYN

Solution : division de T_ENTREPOT par T_RAYON

Implémentation 1 :

Idée : compter le nombre d'occurrences des rayons et de le faire coïncider avec le dénombrement des rayons des différents entrepôts :

```
SELECT VILLE_ETP FROM T_ENTREPOT
    GROUP BY VILLE_ETP
        HAVING COUNT(*) = (SELECT COUNT(*) FROM T_RAYON)
```

Résultat : MARSEILLE, TOULOUSE

Cette solution est **correcte uniquement** si :

1. il n'y a **aucune redondance** de données dans la table diviseur (RYN est unique)
2. il n'y a **pas de rayon en sus pour un entrepôt**, non recensé dans la table T_RAYON

Résolution problème 1 :

```
    … HAVING COUNT(*) = (SELECT COUNT(DISTINCT RYN) FROM T_RAYON)
```

Implémentation 2 (double négation):

```
SELECT DISTINCT VILLE_ETP
      FROM T_ENTREPOT AS ETP1
            WHERE NOT EXISTS
                  (SELECT * FROM T_RAYON RYN
                        WHERE NOT EXISTS
                        (SELECT * FROM T_ENTREPOT AS ETP2
                              WHERE ETP1.VILLE_ETP = ETP2.VILLE_ETP AND
                              (ETP2.RAYON_RYN = RYN.RAYON_RYN)))
```

Cette requête recherche les entrepôts pour qui *il n'existe pas de rayon qu'ils ne peuvent pas fournir*

Implémentation 3 (différence entre tables) :

Solution plus rapide:

```
SELECT DISTINCT VILLE_ETP FROM T_ENTREPOT AS ETP1
      WHERE (SELECT RAYON_RYN FROM T_RAYON
                  EXCEPT
                  SELECT RAYON_RYN FROM T_ENTREPOT AS ETP2
                        WHERE ETP1.VILLE_ETP = ETP2.VILLE_ETP ) IS NULL
```

Implémentation 3 (division exacte) :

Utilisation de la division soit "exacte", c'est à dire que la table dividende corresponde exactement avec les valeurs du diviseur, ni plus ni moins.

→ Trouver quels sont les entrepôts qui disposent de tous les rayons de la table T_RAYON, mais aussi d'aucun autre.
→ Utilisation de la double négation :

```
SELECT VILLE_ETP FROM T_ENTREPOT ETP1
    WHERE NOT EXISTS
    (SELECT * FROM T_RAYON
        WHERE NOT EXISTS
            (SELECT * FROM T_ENTREPOT ETP2
            WHERE (ETP1.VILLE_ETP = ETP2.VILLE_ETP) AND
                    (ETP2.RAYON_RYN = T_RAYON.RAYON_RYN)))
    GROUP BY VILLE_ETP
        HAVING COUNT (*) = (SELECT COUNT(*) FROM T_RAYON)
```

Exemple 2 :

Tables :

PARTICIPER		EPREUVE
Athlète	Epreuve	Epreuve

Les athlètes ayant participé au moins à toutes les épreuves de la table EPREUVE

```
SELECT Athlète FROM PARTICIPER
    GROUP BY Athlète
    HAVING COUNT(*) =
    (SELECT COUNT(DISTINCT Epreuve) FROM EPREUVE) ;
```

Les athlètes qui ont participé uniquement aux épreuves de la table EPREUVE et à aucune autre (Division "exacte"), ici Martin.

→ Basé sur une jointure externe (SQL2) et utilisation la particularité des fonctions d'agrégation d'ignorer les valeurs nulles.

```
SELECT Athlète
FROM PARTICIPER A LEFT JOIN (SELECT DISTINCT Epreuve FROM EPREUVE) B
ON A.Epreuve = B.Epreuve
GROUP BY Athlète
HAVING COUNT(*) = (SELECT COUNT(DISTINCT Epreuve) FROM EPREUVE)
    AND COUNT(B.Epreuve) =
    (SELECT COUNT(DISTINCT Epreuve) FROM EPREUVE) ;
```

Voici un résultat renvoyé par la jointure externe gauche (LEFT JOIN) entre PARTICIPER et EPREUVE :

Athlète	A.Epreuve	B.Epreuve
Dupont	200 m	200 m
Dupont	400 m	400 m
Dupont	110 m H	110 m H
Dupont	100 m	NULL
Martin	200 m	200 m
Martin	400 m	400 m
Martin	110 m H	110 m H
Bertrand	400 m	400 m
Bertrand	200 m	200 m
Michel	200 m	200 m

Ensuite le regroupement avec comptage renvoie :

Athlète	COUNT(*)	COUNT(B.Epreuve)
Dupont	4	3
Martin	3	3
Bertrand	2	2
Michel	1	1

Autre type de solution (plus élégante) basée sur la double négation et les SELECT imbriqués et corrélés :

```
SELECT Athlète FROM PARTICIPER A
WHERE NOT EXISTS (SELECT * FROM EPREUVE
                        WHERE NOT EXISTS
                            (SELECT * FROM PARTICIPER B
                                WHERE (A.Athlète = B.Athlète) AND
                                (B.Epreuve = EPREUVE.Epreuve)))
GROUP BY Athlète
HAVING COUNT(*) = (SELECT COUNT (DISTINCT Epreuve) FROM EPREUVE) ;
```