Compléments SQL sur les requêtes complexes

http://www.lirmm.fr/~poncelet/HLIN605.html

Pascal Poncelet
LIRMM
Pascal.Poncelet@lirmm.fr
http://www.lirmm. fr/~poncelet

LMD - Partitionnement

- Permet de regrouper les tuples d 'une relation en sous classes par valeur de l 'attribut réalisant le partitionnement :

```
GROUP BY col1, [col2, …]
```

- Doit suivre le `WHERE` ou le `FROM` si celui-ci est vide
- Les colonnes mentionnées dans le `GROUP BY` doivent être indiquées dans le `SELECT`
- Attention : si `GROUP BY`, les fonctions agrégatives s 'appliquent aux sous classes

LMD - Partitionnement

- Quel est le nombre de vols effectués par chacun des pilotes ?

SELECT Pl#, **COUNT** (Vol#)
FROM VOL
GROUP BY Pl# ;

Une seule ligne par pilote !

- Nombre de vols par pilote et par avion ?

SELECT Pl#, AV#, **COUNT** (Vol#)
FROM VOL
GROUP BY Pl#, Av# ;

Autant de lignes par pilote qu'il y a d'avions différents qu'il conduit

LMD - Partitionnement

- Pour chaque pilote (num et nom) donner le nombre de vol qu'il assure au départ de Paris

SELECT Pllnom, **COUNT** (*)
FROM PILOTE, VOL
WHERE VD = « PARIS »
AND PILOTE.PL#=VOL.PL#
GROUP BY Plnom;

LMD - Partitionnement

- Donner pour chaque appareil, le nombre pilotes qui l'utilisent.

SELECT NomAv, AVION.V#, Pllnom, **COUNT** (VOL.Pl#)
FROM PILOTE, VOL, AVION
WHERE VOL.PL#=AVION.PL#
AND PILOTE.PL#=VOL.PL#
GROUP BY NomAV, AVION.AV#

LMD - Partitionnement

1	DURAND
2	DUPOND

AV1	AIRBUS
AV2	AIRBUS

V1	1	AV1
V2	2	AV1
V3	1	AV2
V4	2	AV1
V5	2	AV2
V6	1	AV1

A1 : 1 (V1)
2 (V2)
2 (V4)
1 (V6)
= 4

A2 : 1 (V3)
2 (V5)
= 2 (IL N'Y A QUE 2 PILOTES !)

LMD - Partitionnement

- Donner pour chaque appareil, le nombre pilotes qui l'utilisent.

SELECT NomAv, AVION.V#, Pllnom, **COUNT** (**DISTINCT** VOL.Pl#)
FROM PILOTE, VOL, AVION
WHERE VOL.PL#=AVION.PL#
AND PILOTE.PL#=VOL.PL#
GROUP BY NomAV, AVION.AV#

LMD - Partitionnement

- Sélection dans les sous classes
  `GROUP BY ... HAVING` condition
- La condition doit porter sur la sous-table

- Donner le nombre de vols (s'il est supérieur à 5) par pilote

SELECT Plnom, **COUNT** (Vol#)
FROM PILOTE, VOL
WHERE PILOTE.PL# = VOL.PL#
GROUP BY Plnom
HAVING Count(Vol#)>5;

LMD - Partitionnement

- Quels sont les noms des pilotes assurant le même nombre de vols avec un AIRBUS que DUPONT ?
- C'est un partitionnement. Rappel dans le WHERE on manipule l'ensemble donc on ne peut pas tester pilote par pilote
- 2 parties :
- Combien de vols sont faits par DUPONT sur un AIRBUS
- Combien de vols sont égaux à la valeur précédente

LMD - Partitionnement

- Combien de vols sont faits par DUPONT sur un AIRBUS

```
SELECT COUNT (*)
FROM PILOTE, VOL, AVION
WHERE PILOTE.Pl# = VOL.Pl#
AND VOL.Av#=AVION.Av#
AND Plnom = « DUPONT »
AND Nomav LIKE 'AIRBUS%'
```

= RES1

LMD - Partitionnement

- Combien de vols sont égaux à la valeur précédente (RES1)

```
SELECT PILOTE.Pl#, Plnom, COUNT (*)
FROM PILOTE, VOL, AVION
WHERE PILOTE.Pl# = VOL.Pl#
AND VOL.Av#=AVION.Av#
AND Nomav LIKE 'AIRBUS%'
GROUP BY PILOTE.PL#, Plnom
HAVING COUNT (*) = RES1
```

LMD - Partitionnement

- Combien de vols sont égaux à la valeur précédente (RES1)

```
SELECT PILOTE.Pl#, Plnom, COUNT (*)
FROM PILOTE, VOL, AVION
WHERE PILOTE.Pl# = VOL.Pl#
AND VOL.Av#=AVION.Av#
AND Nomav LIKE 'AIRBUS%'
GROUP BY PILOTE.PL#, Plnom
HAVING COUNT (*) = (SELECT COUNT (*)
                    FROM PILOTE, VOL, AVION
                    WHERE PILOTE.Pl# = VOL.Pl#
                    AND VOL.Av#=AVION.Av#
                    AND Plnom = « DUPONT »
                    AND Nomav LIKE 'AIRBUS%')
```

LMD - Partitionnement

- Les seules difficultés sont :
- De ne pas oublier que les colonnes mentionnées dans le `GROUP BY` doivent être indiquées dans le `SELECT`
- Et de ne pas confondre :
  - Les conditions qui sont dans le `WHERE` portent sur l'ensemble de la relation
  - Les conditions qui sont dans le `HAVING` portent sur la sous relation qui a été partitionnée avec le `GROUP BY`

Equivalence algébrique - SQL

- SELECTION

- SELECT(R/critere) ou $\sigma_{(critere)}(R)$
- **SELECT** * **FROM** R **WHERE** critere;

SELECT(PILOTE/ADR=« Paris »)
$\sigma_{(ADR=« PARIS »)}$(PILOTE)

SELECT * **FROM** PILOTE **WHERE** ADR=« PARIS »;

Equivalence algébrique - SQL

- PROJECTION

- PROJECT(R/liste) ou $\Pi_{(liste)}(R)$
- **SELECT** liste **FROM** R;

PROJECT(PILOTE/PLNUM,ADR)
$\Pi_{(PLNUM,ADR)}$(PILOTE)

SELECT PLNUM,ADR **FROM** PILOTE;

Equivalence algébrique - SQL

- PRODUIT CARTESIEN

- ⊗(R,S) ou R ⊗ S
- **SELECT** * **FROM** R,S; (ATTENTION !!!! Ce n'est pas une jointure !!!)

⊗(PILOTE,VOL)
PILOTE ⊗ VOL

SELECT * **FROM** PILOTE, VOL;

Equivalence algébrique - SQL

- JOINTURE
- JOIN(R,S, critere) ou R $\bowtie_{critere}$ S = $\sigma_{(critere)}$(R ⊗ S)
- **SELECT** * **FROM** R,S **WHERE** critere;

JOIN(PILOTE,VOL/PILOTE.PLNUM=VOL.PLNUM)
PILOTE $\bowtie_{PLNUM=PLNUM}$ VOL

SELECT * **FROM** PILOTE,VOL **WHERE** PILOTE.PLNUM=VOL.PLNUM;

Equivalence algébrique - SQL

- UNION
- UNION(R,S) ou R ∪ S
- **SELECT** * **FROM** R **UNION SELECT** * **FROM** S;

UNION(PILOTE1,PILOTE2)
PILOTE1 ∪ PILOTE2

SELECT * **FROM** PILOTE1 **UNION SELECT** * **FROM** PILOTE2;

Equivalence algébrique - SQL

- INTERSECTION
- INTERSECTION(R,S) ou R ∩ S
- **SELECT** * **FROM** R **INTERSECT SELECT** * **FROM** S;
- **SELECT** * **FROM** R **WHERE** A_1, A_2 .. A_N **IN** (**SELECT** * **FROM** S);

INTERSECTION(PILOTE1,PILOTE2)
PILOTE1 ∩ PILOTE2

SELECT * **FROM** PILOTE1 **INTERSECT SELECT** * **FROM** PILOTE2;

Equivalence algébrique - SQL

- DIFFERENCE
- DIFFERENCE(R,S) ou R - S
- **SELECT** * **FROM** R **EXCEPT SELECT** * **FROM** S;
- **SELECT** * **FROM** R **WHERE** A_1, A_2 .. A_N **NOT IN** (**SELECT** * **FROM** S);

DIFFERENCE(PILOTE1,PILOTE2)
PILOTE1 - PILOTE2

SELECT * **FROM** PILOTE1 **EXCEPT SELECT** * **FROM** PILOTE2;

La division en SQL

- Rappel :
- Division d 'une relation binaire par une relation unaire
- R(X,Y) ÷S(Y)
- **Les x associés à tous les y de R**

x1	y1
x1	y2
x2	y1
x2	y3

y1
y2

x1

La division en SQL

- Avions conduits par tous les pilotes VOL1 (PLNUM ÷ PLNUM) PIL ?

VOL1

AVNUM	PLNUM
30	100
30	101
30	102
30	103
31	100
31	102
32	102
32	103
33	102

Diviseur PIL1

PLNUM
100

→

AVNUM
30
31

Diviseur PIL2

PLNUM
102
103

→

AVNUM
30
32

La division en SQL

- Expression en algébrique

$$\Pi_X(R) - \Pi_X(\Pi_X(R) \otimes S - R)$$

- $\Pi_X(R) \otimes S$ = la division idéale : « tous les x associés à tous les y de S »

x1	y1
x1	y2
x2	y1
x2	y2

La division en SQL

- Expression en algébrique

$$\Pi_X(R) - \Pi_X(\Pi_X(R) \otimes S - R)$$

- $\Pi_X(R) \otimes S - R$ = les mauvais (« ceux qui ne sont pas associés à tous les y de S »)

x2	y2

La division en SQL

- Pour avoir les bons :

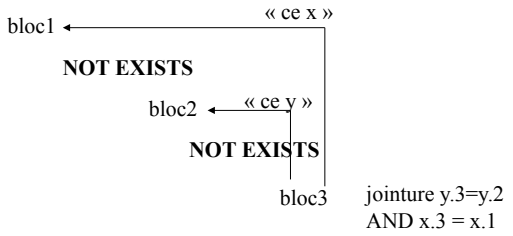
$$\Pi_X(R) - \Pi_X(\Pi_X(R) \otimes S - R)$$

x1
x2

\- | x2 | = | x1 |

La division en SQL

- Utilisation d'un double NOT EXISTS
- **Quels sont les x associés à tous les y de R ?**
- Paraphrase : « Quels sont les x tels qu'il n'existe pas de y qui ne soit pas associé à ce x ? »
- Présence de deux blocs :



La division en SQL

- Les pilotes conduisant tous les avions
- *Existe-t-il un pilote tel qu'il n'existe aucun avion de la compagnie qui ne soit pas conduit par ce pilote ?*

```
SELECT *                                              Bloc 1
FROM PILOTE
WHERE NOT EXISTS (
    SELECT *
    FROM AVION                                        Bloc 2
    WHERE NOT EXISTS (
        SELECT *
        FROM VOL
        WHERE VOL. NUMPIL=PILOTE.NUMPIL               Bloc 3
        AND VOL. NUMAV=AVION.NUMAV));
```


La division en SQL

- Examen de la requête :

- Pour chaque pilote examiné par le 1er bloc, les différents tuples de AVION sont balayés au niveau du 2ème bloc et pour chaque avion, les conditions de jointure du 3ème bloc sont évaluées.

La division en SQL

1	…
2	…
3	…

AV1	….
AV2	….

V1	1	AV1
V2	1	AV1
V3	2	AV2
V4	3	AV1
V5	2	AV1

Considérons le pilote 1. Parcourons les tuples de la relation AVION. Pour l'avion n° 10, le 3ème bloc retourne un résultat (le vol V1), **NOT EXISTS** est donc faux et l'avion AV1 n'appartient pas au résultat du 2ème bloc. L'avion AV2 n'étant jamais piloté par le pilote 1, le 3ème bloc ne rend aucun tuple, le **NOT EXISTS** associé est donc évalué à vrai. Le 2ème bloc rend donc un résultat non vide (l'avion AV2) et donc le **NOT EXISTS** du 1er bloc est faux. Le pilote 1 n'est donc pas retenu dans le résultat de la requête.

La division en SQL

1	…
2	…
3	…

AV1	….
AV2	….

V1	1	AV1
V2	1	AV1
V3	2	AV2
V4	3	AV1
V5	2	AV1

Pour le pilote 2 et l'avion AV1, il existe un vol (V3). Le 3ème bloc retourne un résultat, **NOT EXISTS** est donc faux. Pour le même pilote et l'avion AV2, le 3ème bloc restitue un tuple et à nouveau **NOT EXISTS** est faux. Le 2ème bloc rend donc un résultat vide ce qui fait que le **NOT EXISTS** du 1er bloc est évalue à vrai. Le pilote 2 fait donc partie du résultat de la requête.

La division en SQL

1	…
2	…
3	…

AV1	….
AV2	….

V1	1	AV1
V2	1	AV1
V3	2	AV2
V4	3	AV1
V5	2	AV1

- Pour le pilote 3: comme il ne pilote pas l'avion AV2, le 2ème bloc retourne une valeur et la condition du 1er bloc élimine ce pilote du résultat.

La division en SQL

- Les pilotes conduisant tous les airbus

```
SELECT *
FROM PILOTE                                   Bloc 1
WHERE NOT EXISTS (
    SELECT *
    FROM AVION
    WHERE AVNOM=« AIRBUS »                    Bloc 2
    AND NOT EXISTS (
            SELECT *
            FROM VOL                          Bloc 3
            WHERE VOL. NUMPIL=PILOTE.NUMPIL
            AND VOL. NUMAV=AVION.NUMAV));
```

La condition dans le bloc 2

La division en SQL

- Utilisation d'une partition ou d'un comptage
- **Quels sont les x associés à tous les y de R ?**
- Paraphrase : « Quels sont les x tels que le nombre de y différents auxquels ils sont associés soit égal au nombre total de y ? »

La division en SQL

- Les pilotes conduisant tous les avions
- *Quels sont les pilotes qui conduisent autant d'avions que la compagnie en possède ?*

SELECT NUMPIL
FROM VOL
GROUP BY NUMPIL
HAVING COUNT (**DISTINCT** NUMAV) = **SELECT COUNT**(*) **FROM** AVION;

La division en SQL

SELECT NUMPIL
FROM VOL
GROUP BY NUMPIL
HAVING COUNT (**DISTINCT** NUMAV) = **SELECT COUNT**(*) **FROM** AVION;

- Le comptage dans la clause **HAVING** permet pour chaque pilote de dénombrer les appareils conduits
- L'oubli du **DISTINCT** rend la requête fausse (on compterait alors le nombre de vols assurés)
- Cette technique de paraphrasage ne peut être utilisée que si les deux ensembles dénombrés sont parfaitement comparables

La division en SQL

- Les pilotes conduisant tous les airbus

SELECT NUMPIL
FROM VOL, AVION
WHERE AVION.NUMAV=VOL.NUMAV
AND AVNOM=« AIRBUS »
GROUP BY NUMPIL
HAVING COUNT (**DISTINCT** NUMAV) =
  SELECT COUNT(*)
  FROM AVION
  WHERE AVNOM=« AIRBUS »;

Attention la condition doit être dans les deux