

Le Langage SQL

Langage de manipulation de données (LMD)

HLIN605

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



Introduction

- Origine : développé chez IBM par l'équipe de recherche de Codd - Naissance de Sequel
- Devenu un standard ANSI en 89 (American National Standard Institute)
 - Existence de « différents dialectes » mais les différences de syntaxe sont minimes
- « Aujourd'hui, un SGBDR ne se vend pas sans une interface SQL »



2

Introduction

- Les différentes évolutions

Année	Nom	Appellation	Commentaires
1986	ISO/CEI 9075:1986	SQL-86 ou SQL-87	Édité par l'ANSI puis adopté par l'ISO en 1987.
1989	ISO/CEI 9075:1989	SQL-89 ou SQL-1	Révision mineure.
1992	ISO/CEI 9075:1992	SQL-92 (ex) alias SQL2	Révision majeure.
1999	ISO/CEI 9075:1999	SQL-99 (ex) alias SQL3	Expressions rationnelles, requêtes récursives, déclencheurs, types non-scalaires et quelques fonctions orientées objet (les deux derniers points sont quelque peu controversés et pas encore largement implémentés).
2003	ISO/CEI 9075:2003	SQL:2003 (ex)	Introduction de fonctions pour la manipulation XML, « window functions », ordres standardisés et colonnes avec valeurs auto-produites (y compris colonnes d'identité).
2008	ISO/CEI 9075:2008	SQL:2008 (ex)	Ajout de quelques fonctions de fenêtrage (ntile, lead, lag, first value, last value, nth value), limitation du nombre de ligne (OFFSET / FETCH), amélioration mineure sur les types distincts, curseurs et mécanismes d'auto-incréments.
2011	ISO/CEI 9075:2011	SQL:2011 (ex)	



3

Introduction

- 3 facettes
 - Langage de définition de données (LDD)
 - Création du schéma
 - Langage de manipulation de données (LMD)
 - Mise à jour et interrogation du schéma
 - Langage de contrôle des données (LCD)
 - Autorisations
- SQL est basé sur des mots clés (en anglais) et se veut proche du langage naturel
- Table = Relation ; Colonne = Attribut ; Ligne = tuple



4

Langage de Manipulation (LMD)

- Structure du bloc de base

SELECT {liste des attributs résultats}
Obligatoire

FROM {liste des relations concernées}
Obligatoire

WHERE {liste des conditions} *Facultatif*

;
 Fin de requête



5

LMD - Projections

- Expression des projections
 - Dans la clause SELECT
 - Les différents attributs sont séparés par des virgules (,)
- Liste des villes desservies par la compagnie ?
 - **SELECT** VA **FROM** VOL;
- Noms et Adresses des pilotes ?
 - **SELECT** PINom, Adr **FROM** PILOTE;



6

LMD - Projections

- Attention : avec SQL il n'y a pas d'élimination automatique des duplicats, c'est à l'utilisateur de le spécifier avec la clause **DISTINCT**
SELECT DISTINCT VA FROM VOL ;
- Pour ne pas réaliser de projection, soit on cite tous les attributs de la (les) relations, soit on utilise *
Toutes les informations sur les pilotes
SELECT * FROM PILOTE;



7

LMD - Sélections

- Expression des sélections
 - Dans la clause **WHERE** sous la forme d'une condition :
Attribut θ Constante où $\theta \in \{=, \neq, <, \leq, >, \geq\}$
Quels sont les pilotes Niçois ?
SELECT * FROM PILOTE WHERE Adr = « NICE » ;
- Possibilité d'utiliser des connecteurs logiques **AND**, **OR**, **NOT**. Attention aux priorités (NOT puis AND puis OR) Mettre des parenthèses



8

LMD - Sélections

- Définition d'appartenance à un intervalle
WHERE att BETWEEN borneinf AND bornesup
- Appartenance à une liste
WHERE att IN (val1, val2, ..., valn) avec att=val1 ou att = val2 ou ... att = valn
- Recherche de sous chaînes
WHERE att LIKE « chaîne générique »
_ pour un caractère quelconque ou % pour une chaîne. Sous Unix _ = ? et % = *.
- **NULL**
WHERE att IS NULL



9

LMD - Sélections

- Donner tous les informations sur les Airbus dont le numéro est compris entre 100 et 150 et qui sont localisés à NICE, MARSEILLE, TOULOUSE ou BORDEAUX

```
SELECT * FROM AVION
WHERE Avnom LIKE « Airbus% »
AND Avnum BETWEEN 100 AND 150
AND Loc IN (« NICE », « MARSEILLE »,
« TOULOUSE », « BORDEAUX »);
```



10

LMD - Sélections

- Donner tous les informations sur les pilotes qui n'ont pas de salaire

```
SELECT *
FROM PILOTE
WHERE sal IS NULL;
```



11

LMD - Calculs verticaux

- Dans un mapping, il est possible d'utiliser des fonctions agrégatives, appliquées sur les valeurs d'attributs. Ces fonctions sont :
Sum, Avg, Min, Max, Stddev, Variance, Count
– Quel est le total des salaires des pilotes ?
SELECT SUM(Sal) FROM PILOTE ;
- Utiliser **DISTINCT** si vous ne voulez pas que le calcul se fasse sur les duplicats !
- COUNT** admet * comme argument = rend le nombre de tuples sélectionnés



12

LMD - Calculs verticaux

- Exemples
- Quel est le nombre de villes desservies par la compagnie ?
SELECT COUNT (DISTINCT VA) FROM VOL ;
- Quel est le nombre de Vols à destination de Nice ?
SELECT COUNT (*) FROM VOL WHERE VA = « NICE »;



13

LMD - Calculs verticaux

- Renommer le résultat : **AS**
SELECT COUNT (DISTINCT VA) FROM VOL ;

COUNT (DISTINCT VA)
PARIS
NICE

SELECT COUNT (DISTINCT VA) AS VILLES FROM VOL ;

VILLES
PARIS
NICE
- Ne modifie rien dans les tables. Renomme uniquement le résultat



14

LMD - Calculs verticaux

- Attention le résultat d'une fonction d'agrégation retourne une seule valeur :

SELECT Numpil, Count(*)
FROM VOL;
- Impossible** car on essaye d'associer à chaque valeur de Numpil le nombre de vols



15

LMD - Calculs horizontaux

- Calculs en utilisant :
 - des opérateurs : +, -, *, / et || (concaténation de chaînes)
 - des fonctions : ABS, SQRT, COS, ...

Quels sont les noms des pilotes qui avec une augmentation de 10% de leur salaire gagnent moins de 2000 € ?

```
SELECT P1nom
FROM PILOTE
WHERE Sal * 1.1 < 2000;
```



16

LMD - Jointures prédictives

- Dans la clause WHERE sous forme
att1 θ att2 où $\theta \in \{=, \neq, <, \leq, >, \geq\}$
- Si les attributs de jointure portent le même nom,
préfixer par le nom de la relation
Donner les numéros et horaires des vols au départ de
Paris assurés par un A320 ?

```
SELECT Volnum, HD, HA
FROM VOL, AVION
WHERE VOL.Avnum = AVION.Avnum
AND VD = « PARIS »
AND Avnom = « A320 » ;
```



17

LMD - Autojointures

- Lorsque l'on utilise 2 fois la même relation dans
un bloc, utiliser des alias ou synonymes pour
différencier les rôles joués par la relation

Numéros des pilotes gagnant le même salaire que
Dupont ?

```
SELECT P1.Plum FROM PILOTE P1, PILOTE P2
WHERE P1.Sal = P2. Sal
AND P2.Plum = « Dupont »
AND P1.Plum <> P2.Plum;
```



18

LMD - Jointures imbriquées

- Sous Requêtes
- 1er cas : Le résultat de la sous requête est une unique valeur => utiliser un opérateur de comparaison entre les 2 blocs

Nom des pilotes qui gagnent plus que la moyenne

```
SELECT Plnom FROM PILOTE
WHERE Sal > (SELECT AVG(Sal)
             FROM PILOTE);
```



19

LMD - Jointures imbriquées

- 2nd cas : le résultat de la sous requête est un ensemble de valeurs
 - Si la condition doit être vérifiée pour une des valeurs de la liste, on fait précéder la sous requête de **IN** ou **=ANY**

Nom des pilotes assurant un vol au départ de Nice

```
SELECT Plnom FROM PILOTE
WHERE Plnum IN (SELECT Plnum
                FROM VOL
                WHERE VD = « NICE »);
```



20

LMD - Jointures imbriquées

- 2nd cas : le résultat de la sous requête est un ensemble de valeurs
 - Si la condition doit être vérifiée pour toutes les valeurs de la liste, on fait précéder la sous requête de **ALL** où **θ** est un opérateur de comparaison

Noms des pilotes Niçois qui gagnent plus que les pilotes Parisiens ?

```
SELECT Plnom FROM PILOTE
WHERE ADR = « NICE »
AND Sal > ALL (SELECT DISTINCT Sal
               FROM PILOTE
               WHERE ADR = « PARIS »);
```



21

LMD - Jointures imbriquées

- Possibilités de travailler sur un ensemble d'attributs

Quels sont les avions de même nom et localisés au même endroit que l'avion Numéro 105 ?

```
SELECT *
FROM AVION
WHERE (Avnom, Loc) = (SELECT Avnom, Loc
                      FROM AVION
                      WHERE Avnum = 105);
```

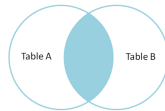


22

Discussions sur les jointures

- Jointure Interne :

```
SELECT Volnum, HD, HA FROM VOL, AVION
WHERE VOL.Avnum = AVION.Avnum
AND VD = « PARIS »
AND Avnom = « A320 » ;
```



- Autre syntaxe :

```
SELECT Volnum, HD, HA FROM VOL
INNER JOIN AVION ON VOL.Avnum = AVION.Avnum
WHERE VD = « PARIS »
AND Avnom = « A320 » ;
```

2 tables

```
SELECT Volnum, HD, HA FROM VOL
INNER JOIN AVION ON VOL.Avnum = AVION.Avnum
INNER JOIN PILOTE ON VOL.Plnum=PILOTE.plnum
WHERE VD = « PARIS »
AND Avnom = « A320 » ;
```

3 tables



23

Discussions sur les jointures

- En fait il existe différents types de jointures
- Le choix de la jointure dépend de ce que l'on recherche



24

Discussions sur les jointures

- Jointure Externe :
- Vouloir récupérer un résultat même s'il n'y a pas de valeurs associées (champs NULL)
- Syntaxe :
LEFT | RIGHT | FULL OUTER JOIN
 table_de_jointure **ON** condition



25

Discussions sur les jointures

- Exemple : il existe des vols qui partent de NICE mais un seul pilote associé et on souhaite avoir tous les pilotes même sans vol
SELECT Plnum, Plnom, HD, HA **FROM** VOL
LEFT OUTER JOIN PILOTE **ON** VOL.Plnum = PILOTE.Plnum
WHERE VD = « NICE »
AND Avnom = « A320 » ;

Plnum	Plnom	HD	HA
1	DUPONT	NULL	NULL
2	DURAND	NULL	NULL
3	DUJARDIN	12	18



26

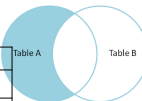
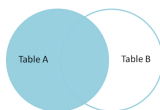
Discussions sur les jointures

• LEFT JOIN

Intérêt permet de reporter tous les tous les résultats de la relation de gauche même s'il n'y a pas de correspondance dans table de droite

SELECT Plnum, Plnom, HD, HA **FROM** VOL
LEFT OUTER JOIN PILOTE **ON** VOL.Plnum = PILOTE.Plnum
WHERE HD IS NULL;

Plnum	Plnom	HD	HA
1	DUPONT	NULL	NULL
2	DURAND	NULL	NULL

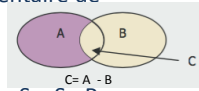


27

RAPPEL DIFFERENCE

- R - S : ensemble des tuples qui appartiennent à R sans appartenir à S. Complémentaire de l'intersection :

$$R - S = \{t / t \in R \text{ ET } t \notin S\}$$



- Opérateur non commutatif : $R - S \neq S - R$

PILOTE1 – PILOTE2	PLNUM	ADR
	100	PARIS
	120	PARIS

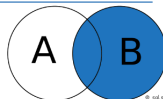
PILOTE 1 – PILOTE 2 : ensemble des pilotes habitant PARIS et n'assurant pas de vol au départ de PARIS ou TOULOUSE



Discussions sur les jointures

- RIGHT JOIN

Intérêt permet de reporter tous les tous les résultats de la relation de droite même s'il n'y a pas de correspondance dans la table de gauche



SELECT Plnom, HD, HA **FROM** VOL
RIGHT OUTER JOIN PILOTE **ON** VOL.Plnum = PILOTE.Plnum
WHERE Plnom **IS** NULL;

Plnom	HD	HA
NULL	12	14

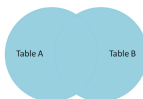


29

Discussions sur les jointures

- FULL OUTER JOIN

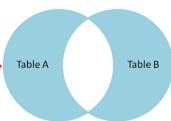
Intérêt permet de reporter tous les tous les résultats de la relation de droite et de gauche même s'il n'y a pas de correspondance
 Un NULL est attribué à droite ou à gauche



SELECT * FROM PILOTE **FULL OUTER JOIN** PILOTE.Plnum = VOL.Plnum
WHERE
 PILOTE.Plnum **IS** NULL **OR** VOL.Plnum **IS** NULL;

- Affichera :

NULL 12 14
 1 NULL



30

LMD - Op. ensemblistes

- Syntaxe :
SFW {UNION, INTERSEC, MINUS} SFW
- Contraintes : uncompatibilité des relations opérandes - Attention à l'ordre => compatibilité syntaxique des attributs projetés dans les deux clauses SELECT
- Pas forcément supportés par tous les SGBD !
- Equivalence avec d'autres opérations d'interrogation
- MINUS = EXCEPT



31

LMD - Op. ensemblistes

- Equivalence avec d'autres opérations d'interrogation

SELECT PInom **FROM** PILOTE **WHERE** Adr=« NICE »
UNION

SELECT PInom **FROM** PILOTE **WHERE** Adr = « PARIS » ;

SELECT PInom **FROM** PILOTE **WHERE** Adr = « NICE » **OR**
Adr = « PARIS » ;

SELECT PInom **FROM** PILOTE **WHERE** Adr **IN** (« PARIS »,
« NICE »)



32

LMD - Op. ensemblistes

- Equivalence avec d'autres opérations d'interrogation
- Numéros des pilotes qui habitent à NICE et dont la ville de départ d'un vol est PARIS

SELECT PInum **FROM** PILOTE **WHERE** Adr=« NICE »
INTERSECT

SELECT PInum **FROM** PILOTE **WHERE** VD = « PARIS » ;

SELECT PInum **FROM** PILOTE
WHERE PInum **IN** (**SELECT** **DISTINCT** plnum
FROM VOL
WHERE VD = « PARIS ») ;



33

LMD - Op. ensemblistes

- Numéros des pilotes habitant Nice et n'assurant aucun vol au départ de Nice

```
SELECT Pnum FROM PILOTE WHERE Adr = « NICE »
MINUS
SELECT DISTINCT Pnum FROM VOL WHERE VD=« NICE » ;
```

```
SELECT Pnum FROM PILOTE
WHERE Adr = « NICE »
AND Pnum NOT IN (SELECT DISTINCT Pnum
FROM VOL
WHERE VD =« NICE »);
```



34

LMD - Op. ensemblistes

- Attention le produit cartésien existe !

```
SELECT * FROM PILOTE, VOL;
```

- Il y a un résultat qui donne le produit cartésien de PILOTE x VOL !
- C'est un produit cartésien et non pas une jointure !



35

LMD - Tri des résultats

- Il est possible d'ordonner les résultats en SQL, ordre croissant ou décroissant sur un ou plusieurs attributs en utilisant la clause

ORDER BY expression [ASC, DESC], ... où expression est un attribut ou ensemble d'attributs spécifiés dans le SELECT

- Dernière clause du bloc
- Si plusieurs expressions, d'abord tri sur le 1er, puis le 2nd, ...
Liste des pilotes Niçois par ordre de salaire décroissant puis par ordre alphabétique des noms

```
SELECT Pnom, Sal
FROM PILOTE
WHERE Adr = « NICE »
ORDER BY Sal DESC, Pnom ;
```



36

LMD - Mise à Jour

- Modification de la valeur d'un attribut
UPDATE nomrelation **SET** att1=val1, att2=val2 ...
 * condition
- Si absence de conditions, il y a une modification sur tous les tuples de la relations concernées
- La nouvelle valeur peut être fonction de l'ancienne ou être le résultat d'une requête
- Des jointures peuvent être exprimées dans la clause WHERE mais une seule relation est spécifiée dans la clause UPDATE



37

LMD - Mise à Jour

- Exemples
- Le pilote Dupond change d'adresse et son salaire est augmenté de 10 %
UPDATE PILOTE **SET** ADR=« PARIS », Sal=Sal*1.1 **WHERE** Pnom = « Dupont »
- Le pilote de numéro 105 a maintenant le même salaire que le pilote numéro 110
UPDATE PILOTE **SET** Sal = (SELECT Sal
 FROM PILOTE
 WHERE Pnum=110)
WHERE Pnum=105;



38

LMD - Mise à Jour

- Insertion d'un tuple
INSERT INTO nomrelation (list_att) **VALUES** (list_val) ;
- Si la liste des attributs n'est pas spécifiée, il faut donner les valeurs pour chacun des attributs de la relation dans l'ordre de création
- On peut utiliser le mot clé NULL si l'attribut n'a pas de valeur



39

LMD - Mise à Jour

- Insertion d'un nouveau pilote

INSERT INTO PILOTE (Pnum, Plnom, Adr, Sal) **VALUES** (206, « Dupond », « MONTPELLIER », 3000) ;

Remarque : si le pilote 206 existe déjà étant donné que Pnum est clé primaire il ne pourra pas être inséré

INSERT INTO PILOTE (Pnum, Plnom) **VALUES** (207, « Durand ») ;

207 Durand NULL NULL



40

LMD - Mise à Jour

- Suppression de tuples

DELETE FROM nomrelation **WHERE** condition

- Une seule relation dans le FROM

- **DELETE FROM** PILOTE **WHERE** Pnum=206;



41

LMD - Mise à Jour

- Attention : Une opération de mise à jour n'est pas inscrite définitivement dans la base après son exécution

- Notion de transactions (voir en cours plus tard)



42

Vers les requêtes complexes

- Il est possible de vouloir traiter des tuples par sous ensemble : partitionnement
- Il est possible d'avoir des contraintes d'existences
- Il est possible d'avoir des conditions qui ne s'appliquent pas à l'ensemble des tuples
- La requête principale et la sous requête ne sont pas indépendantes



43

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



44

LMD - Partitionnement

- Sans partitionnement : ensemble des tuples
- SELECT** Volnum, Plnum, Avnum, VD, VA
FROM VOL;

Volnum	Plnum	Avnum	VD	VA
100	1	10	PARIS	MONTPELLIER
101	1	11	NICE	LYON
102	2	13	LILLE	NANTES
103	1	10	PARIS	NANTES
104	1	15	LYON	LILLE
105	4	12	PARIS	NICE
106	1	13	NANTES	LILLE



45

LMD - Partitionnement

- Sans partitionnement : ensemble des tuples

SELECT Pnum
FROM VOL;

Pnum
1
1
2
1
1
4
1



46

LMD - Partitionnement

SELECT Pnum
FROM VOL
GROUP BY Pnum;

3
Partitions

Pnum
1
1
1
1
1
2
4



47

On regroupe
par rapport aux numéros de
pilotes

La base de données est
partitionnée
et on analyse chaque partition
à la fois

LMD - Partitionnement

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

On veut connaître par pilote le nombre de vols qu'il a effectué. Une requête sans partitionnement considère l'ensemble de la base

Volnum	Pnum	Avnum	VD	VA
100	1	10	PARIS	MONTPELLIER
101	1	11	NICE	LYON
102	2	13	LILLE	NANTES
103	1	10	PARIS	NANTES
104	1	15	LYON	LILLE
105	4	12	PARIS	NICE
106	1	13	NANTES	LILLE




48

LMD - Partitionnement

- Quel est le nombre de vols effectués par chacun des pilotes ?
SELECT Plnum, **COUNT** (Volnum)
FROM VOL
GROUP BY Plnum ;

On effectue une partition et ensuite on compte le nombre de vols par partition

- Les fonctions agrégatives s'appliquent aux partitions



49

LMD - Partitionnement


SELECT Plnum, **COUNT** (Volnum)
FROM VOL
GROUP BY Plnum ;

COUNT (VOLNUM)
Par partition
Pour Plnum=1
COUNT(VOLNUM)=4

Pour Plnum=2
COUNT(VOLNUM)=1

Pour Plnum=4
COUNT(VOLNUM)=1

Volnum	Plnum	Avnum	VD	VA
100	1	10	PARIS	MONTPELLIER
101	1	11	NICE	LYON
103	1	10	PARIS	NANTES
104	1	15	LYON	LILLE
106	1	13	NANTES	LILLE
102	2	13	LILLE	NANTES
105	4	12	PARIS	NICE




50

LMD - Partitionnement

SELECT Plnum, **COUNT** (Volnum)
FROM VOL
GROUP BY Plnum ;

Plnum	COUNT(Volnum)
1	4
2	1
4	1




51

LMD - Partitionnement

```
SELECT Plnum,
COUNT (Volnum) AS « Nombre de vols »
FROM VOL
GROUP BY Plnum ;
```

Plnum	Nombre de vols
1	4
2	1
4	1



52


LMD - Partitionnement

- Nombre de vols par pilote et par avion ?

```
SELECT Plnum, Avnum, COUNT (Volnum)
FROM VOL
GROUP BY Plnum, Avnum;
```

Partition
Par Plnum

Volnum	Plnum	Avnum	VD	VA
100	1	10	PARIS	MONTPELLIER
101	1	11	NICE	LYON
103	1	10	PARIS	NANTES
104	1	15	LYON	LILLE
106	1	13	NANTES	LILLE
102	2	13	LILLE	NANTES
105	4	12	PARIS	NICE



53


LMD - Partitionnement

- Nombre de vols par pilote et par avion ?

```
SELECT Plnum, Avnum, COUNT (Volnum)
FROM VOL
GROUP BY Plnum, Avnum;
```

Partition
par Plnum et
par Avnum

Volnum	Plnum	Avnum	VD	VA
100	1	10	PARIS	MONTPELLIER
103	1	10	PARIS	NANTES
101	1	11	NICE	LYON
104	1	15	LYON	LILLE
106	1	13	NANTES	LILLE
102	2	13	LILLE	NANTES
105	4	12	PARIS	NICE



54


LMD - Partitionnement

- Nombre de vols par pilote et par avion ?

```
SELECT Plnum, Avnum, COUNT (Volnum)
FROM VOL
GROUP BY Plnum, Avnum;
```

Plnum	Avnum	COUNT(Volnum)
1	10	2
1	11	1
1	15	1
1	13	1
2	13	1
4	12	1

1 ligne par numéro de pilote et numéro d'avion différent




55

LMD - Partitionnement

- Pour chaque pilote (nom) donner le nombre de vol qu'il assure au départ de Paris
- La condition peut porter sur l'ensemble des tuples de la base et non pas sur la partition

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




56

LMD - Partitionnement

- Donner pour chaque appareil, le nombre de pilotes qui l'utilise

```
SELECT AVION.NumAvnum,
       NomAv,
       COUNT (VOL.Plnum)
FROM PILOTE, VOL, AVION
WHERE VOL.Plnum=AVION.Plnum
AND PILOTE.Plnum=VOL.Plnum
GROUP BY AVION.Avnum, NomAV
```



57

LMD - Partitionnement

1	DURAND	AV1	AIRBUS	V1	1	AV1
2	DUPOND	AV2	AIRBUS	V2	2	AV1
				V3	1	AV2
				V4	2	AV1
				V5	2	AV2
				V6	1	AV1

AV1 : P1 (V1) AV2 : P1 (V3)
 P2 (V2) P2 (V5)
 P2 (V4)
 P1 (V6)



= 4 = 2 (IL N'Y A QUE 2 PILOTES !)

58

LMD - Partitionnement

- Donner pour chaque appareil, le nombre de pilotes qui l'utilise

```
SELECT AVION.NumAvnum,
       NomAv,
       COUNT (DISTINCT VOL.Plnum)
FROM PILOTE, VOL, AVION
WHERE VOL.Plnum=AVION.Plnum
AND PILOTE.Plnum=VOL.Plnum
GROUP BY AVION.Avnum, NomAV
```



59

LMD - Partitionnement

- Condition pour la partition

```
GROUP BY ...
HAVING condition
```



60

LMD - Partitionnement

- Donner par pilote le nombre de vols (s'il est supérieur à 5)
SELECT Plnum, **COUNT** (Volnum)
FROM PILOTE, VOL
WHERE PILOTE.Plnum = VOL.Plnum
GROUP BY Plnum
HAVING Count(Volnum)>5;
- Ne retourner un résultat que si le nombre de vol pour un pilote est supérieur à 5.



61

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



62

LMD - Partitionnement

- Combien de vols sont faits par DUPONT sur un AIRBUS
SELECT COUNT (*)
FROM PILOTE, VOL, AVION
WHERE PILOTE.Plnum = VOL.Plnum
AND VOL.Avnum=AVION.Avnum
AND Plnom = « Dupont »
AND Nomav LIKE 'AIRBUS%'

= RES1



63

LMD - Partitionnement

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

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



64

LMD - Partitionnement

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

```
SELECT PILOTE.Plnum, Plnom, COUNT (*)
FROM PILOTE, VOL, AVION
WHERE PILOTE.Plnum = VOL.Plnum
AND VOL.Avnum=AVION.Avnum
AND Nomav LIKE 'AIRBUS%'
GROUP BY PILOTE.Plnum, Plnom
HAVING COUNT (*) = (SELECT COUNT (*)
FROM PILOTE, VOL, AVION
WHERE PILOTE.Plnum = VOL.Plnum
AND VOL.Avnum=AVION.Avnum
AND Plnom = « Dupont »
AND Nomav LIKE 'AIRBUS%')
```



65

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**



66

LMD – Requêtes corrélées

- Jusqu'à présent il n'y avait pas de corrélations entre les requêtes et les sous requêtes

```
SELECT *
FROM PILOTE
WHERE Sal = (SELECT Sal
             FROM PILOTE
             WHERE Pnum=110)
```

Exécution de la sous requête
qui donne un résultat le
salaire du pilote 110



67

LMD – Requêtes corrélées

- Existe-t'il des pilotes n'ayant fait aucun vol ?

```
SELECT *
FROM PILOTE LesPilotes
WHERE NOT EXISTS (SELECT *
                  FROM Vol
                  WHERE LesPilotes.Pnum=Vol.Pnum)
```

Il ne doit pas exister
de pilotes dans la
jointure – Pas de
résultat

On regarde par rapport aux
pilotes qui sont parcourus
dans la requête principale

68

LMD – Requêtes corrélées

- Quels sont les pilotes qui effectuent des vols ?

```
SELECT *
FROM PILOTE LesPilotes
WHERE EXISTS (SELECT *
              FROM Vol
              WHERE LesPilotes.Pnum=Vol.Pnum)
```

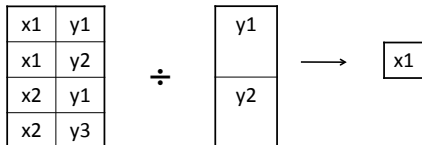
Les pilotes doivent
exister dans la
jointure – Il y a un
résultat

On regarde par rapport aux
pilotes qui sont parcourus
dans la requête principale

69

La division en SQL

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



70

DIVISION - Rappel algébrique

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

VOL1 Dividende	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	AVNUM
		103	30
			32

71

La division en SQL

- Equivalence 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

72

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
----	----



73

La division en SQL

- Pour avoir les bons :

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

x1	-	x2	=	x1
x2				



74

La division en SQL

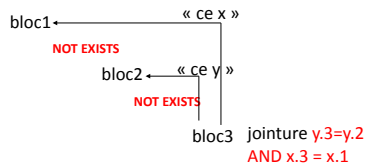
- Quels sont les x associés à tous les y de R ?
- Paraphrase : « Quels sont les x tels qu'il n'existe pas de y qui ne soit pas associé à ce x ? »
- Utilisation d'un double NOT EXISTS



75

La division en SQL

- « Quels sont les x tels qu'il n'existe pas de y qui ne soit pas associé à ce x ? »
- Présence de deux blocs :



76

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
                  WHERE NOT EXISTS (SELECT *
                                    FROM VOL
                                    WHERE VOL. NUMPIL=PILOTE.NUMPIL
                                    AND VOL. NUMAV=AVION.NUMAV));

```

Bloc 2

Bloc 3



77

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

Pnum	...	Avnum	...	Volnum	Pnum	Avnum
1	...	10	...	V1	1	10
2	...	20	...	V2	1	10
				V3	2	10
				V4	2	20

Pour le pilote 1

Parcours des 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 10 n'appartient pas au résultat du 2ème bloc.

L'avion 20 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 20) et donc le **NOT EXISTS** du 1er bloc est faux.

Le pilote 1 n'est donc pas retenu dans le résultat de la requête. 79

La division en SQL

Pnum	...	Avnum	...	Volnum	Pnum	Avnum
1	...	10	...	V1	1	10
2	...	20	...	V2	1	10
				V3	2	10
				V4	2	20

Pour le pilote 2

avec l'avion 10, il existe un vol (V3)

Le 3ème bloc retourne un résultat, **NOT EXISTS** est donc faux.

avec l'avion 20, 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 évalué à vrai.

Le pilote 2 fait donc partie du résultat de la requête.

La division en SQL

- Les pilotes conduisant tous les airbus

Bloc 1

SELECT *

FROM PILOTE

WHERE NOT EXISTS (SELECT *

Bloc 2

FROM AVION

WHERE Avnum=« AIRBUS »

AND NOT EXISTS (SELECT *

Bloc 3

FROM VOL

WHERE VOL. Pnum=PILOTE.Pnum

AND VOL.Avnum=AVION.Avnum));

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 ? »



82

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 Pnum
FROM VOL
GROUP BY Pnum
HAVING COUNT (DISTINCT Avnum) = (SELECT COUNT(*)
                                FROM AVION);
```



83

La division en SQL

```
SELECT Pnum
FROM VOL
GROUP BY Pnum
HAVING COUNT (DISTINCT Avnum) = (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



84

La division en SQL

- Les pilotes conduisant tous les airbus

```
SELECT Pnum
FROM VOL, AVION
WHERE AVION.Avnum=VOL.Avnum
AND Avnom=« AIRBUS »
GROUP BY Pnum
HAVING COUNT (DISTINCT Avnum) =
    (SELECT COUNT(*)
     FROM AVION
     WHERE Avnom=« AIRBUS »);
```



Attention la condition doit être dans les deux

85

- Des questions ?



86
