

Le Langage SQL

Langage de manipulation de données (LMD)

HLIN511

Pascal Poncelet
Pascal.Poncelet@umontpellier.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 (sa) alias SQL-2	Révision majeure.
1999	ISO/CEI 9075:1999	SQL-99 (sa) alias SQL-3	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 (sa)	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 (sa)	Ajout de quelques fonctions de fenêtrage (mile, 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 (sa)	



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 Pnom,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 - Projections

- Le coût d'un **DISTINCT**
- Est ce que ces deux requêtes sont équivalentes ?
SELECT Pnom FROM PILOTE;
SELECT DISTINCT Pnom FROM PILOTE;
- La première affiche l'ensemble des valeurs sans traitement donc les éventuels duplicats
- La seconde nécessite de trier la relation PILOTE par rapport aux différentes valeurs de Pnom et ensuite élimine les duplicats. Peut être une opération coûteuse lorsqu'il y a beaucoup de tuples.



8

LMD - Sélections

- Expression des sélections
 - Dans la clause **WHERE** sous la forme d'une condition :
Attribut X Constante où $X = \{=, >=, <=, <>\}$
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



9

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 OR att = val2
OR ... 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



10

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');
```



11

LMD - Sélections

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

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



12

LMD - Calculs verticaux

- Dans un bloc, il est possible d'utiliser des fonctions agrégatives, appliquées sur les valeurs d'attributs. Ces fonctions sont :
SUM, AVG, MIN, MAX, COUNT, etc..
– 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. Il rend le nombre de tuples sélectionnés



13

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';



14

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.
- Mettre des " " si la chaîne contient des espaces :
SELECT COUNT (DISTINCT VA) AS "Les Villes" FROM VOL ;



15

LMD - Calculs verticaux

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

```
SELECT PInum, COUNT(*)
FROM VOL;
```

*ERROR at line 1:ORA-00937: not a single-group group function

- Impossible** car on essaye d'associer à chaque valeur de PInum le nombre de vols



16

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 PInom
FROM PILOTE
WHERE Sal * 1.1 < 2000;
```



17

LMD - Jointures prédictives

- Dans la clause WHERE sous forme att1 X att2 où X = {=, >=, <=, <>}
- 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';
```



18

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.Plnum
FROM PILOTE P1, PILOTE P2
WHERE P1.Sal = P2. Sal
AND P2.Plnom = 'DUPONT'
AND P1.Plnum <> P2.Plnum;
```

Elimination du pilote Dupont dans le résultat

19

LMD - Jointures imbriquées

- Sous requêtes ou requêtes imbriquées
- 1er cas : Le résultat de la sous requête est une unique valeur
- utiliser un opérateur de comparaison entre les deux blocs

Nom des pilotes qui gagnent plus que la moyenne

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

20

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');
```

21

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 θ 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');

Cette requête n'est pas la plus efficace bien sûr.

Celle-ci est plus appropriée :

SELECT max(Sal)

FROM PILOTE

WHERE Adr = 'PARIS'



22

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);



23

Discussions sur les jointures

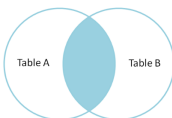
- Jointure Interne :

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



24

Discussions sur les jointures

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



25

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



26

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** PILOTE.Plnum, Plnom, HD, HA **FROM** PILOTE
LEFT OUTER JOIN PILOTE **ON** VOL.Plnum = PILOTE.Plnum
WHERE VD = 'NICE' ;

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




27

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 PILOTE.Plnum, Plnom, HD, HA **FROM** PILOTE
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

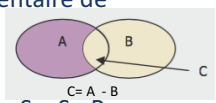
RAPPEL DIFFERENCE

- 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

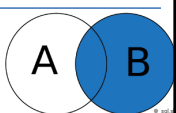
PILOTE 1 – PILOTE 2 : ensemble des pilotes habitant PARIS et
n'assurant pas de vol au départ de PARIS ou TOULOUSE

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

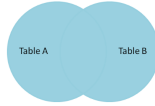


Plnom	Sal	HD	HA
DUPONT	NULL	18	19
DURAND	NULL	22	23
DUJARDIN	50 000	12	18

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 PILOTE.Plnum, Volnum FROM PILOTE
FULL OUTER JOIN VOL ON
```

```
PILOTE.Plnum = VOL.Plnum;
```

Affichera toutes les valeurs même s'il n'y en a pas.

LMD - Op. ensemblistes

- Syntaxe :
SFW {**UNION**, **INTERSEC**, **EXCEPT**} SFW
- Contraintes : unicompatibilité des relations
opérandes - Attention à l'ordre : compatibilité
syntaxique des attributs projetés dans les deux
clauses **SELECT**
- Pas forcément supportés par tous les SGBD !
- **MINUS** = **EXCEPT** (depuis SQL 2002)



32

LMD - Op. ensemblistes

- Equivalence avec d'autres opérations d'interrogation

Liste des pilotes qui habitent NICE et PARIS

```
SELECT Plnom FROM PILOTE WHERE Adr='NICE'
UNION
SELECT Plnom FROM PILOTE WHERE Adr = 'PARIS';
```

```
SELECT Plnom
FROM PILOTE
WHERE Adr = 'NICE' OR Adr = 'PARIS';
```

```
SELECT Plnom
FROM PILOTE
WHERE Adr IN ('PARIS', 'NICE');
```



33

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 Pnum FROM PILOTE WHERE Adr='NICE'
INTERSECT
SELECT Pnum FROM PILOTE WHERE VD = 'PARIS' ;
```

```
SELECT Pnum FROM PILOTE
WHERE Pnum IN (SELECT DISTINCT Pnum
FROM VOL
WHERE VD = 'PARIS') ;
```



34

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');
```



35

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 !



36

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 ;



37

LMD – Le FROM

- Le **FROM** contient les relations sur lesquelles vont s'effectuer les opérations dans le bloc

SELECT VA **FROM** VOL;

SELECT Avion.Avnum, Volnum, HD, HA
FROM VOL, AVION
WHERE VOL.Avnum = AVION.Avnum
AND AVION.Avnum = 100;



38

LMD – Le FROM

- Il est possible de renommer une relation

SELECT A.Avnum, Volnum, HD, HA
FROM VOL V, AVION A
WHERE V.Avnum = A.Avnum
AND A.Avnum = 100;

Avec MySQL possibilité d'écrire

FROM VOL **AS** V, AVION **AS** A
FROM TABLE VOL **AS** V, AVION **AS** A

Ne fonctionne pas sous ORACLE



39

LMD – Le FROM

- Attention quand une relation est renommée il n'est pas possible d'utiliser l'ancien nom

```
SELECT A.Avnum, Volnum, HD, HA
FROM VOL V, AVION A
WHERE V.Avnum = A.Avnum
AND AVION.Avnum = 100 ;
```



FAUX
AVION ne peut plus être
utilisé

40

LMD – Le FROM

- Intérêt : ajout de plus de sémantique
Numéros des pilotes gagnant le même salaire que DUPONT ?

```
SELECT P1.Plnum
FROM PILOTE LesPilotes, PILOTE LePiloteDupont
WHERE LesPilotes.Sal = LePiloteDupont. Sal
AND LePiloteDupont.Plnom = 'DUPONT'
AND LesPilotes.Plnum <> LePiloteDupont.Plnum;
```



- Il est clair que dans les relations, **LePiloteDupont** référence une relation particulière. Celle pour laquelle on veut comparer les salaires de tous les pilotes (**LesPilotes**)

41

LMD – Le FROM

- Le contenu d'un **FROM** peut être une requête

```
SELECT A.Avnum, Volnum, HD, HA
FROM (SELECT * FROM VOL) V, AVION A
WHERE V.Avnum = A.Avnum
AND A.Avnum = 100;
```



42

LMD - Mise à Jour

- Modification de la valeur d'un attribut

UPDATE nomrelation **SET** att1=val1, att2=val2 ... {**WHERE** Condition}

- Si absence de conditions (pas de clause **WHERE**), 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**



43

LMD - Mise à Jour

- Exemples
- Le pilote DUPONT change d'adresse et son salaire est augmenté de 10 %

UPDATE PILOTE **SET** Adr='PARIS', Sal=Sal*1.1
WHERE Plnom = '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 Plnum=110)
WHERE Plnum=105;



44

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



45

LMD - Mise à Jour

- Insertion d'un nouveau pilote

INSERT INTO PILOTE (PInum, PInom, Adr, Sal)

VALUES (206, 'DUPOND', 'MONTPELLIER', 3000) ;

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

INSERT INTO PILOTE (PInum, PInom) **VALUES** (207, 'DURAND') ;
<207, 'DURAND', NULL NULL>



46

LMD - Mise à Jour

- Suppression de tuples

DELETE FROM nomrelation **WHERE** condition

- Une seule relation dans le FROM

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



47

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)



48

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



49

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



50

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



51

LMD - Partitionnement

- Sans partitionnement : ensemble des tuples

SELECT Pnum

FROM VOL;

Pnum
1
1
2
1
1
4
1



52

LMD - Partitionnement

SELECT Pnum

FROM VOL

GROUP BY Pnum;

On regroupe
par rapport aux numéros de
pilotes

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

3
Partitions

Pnum
1
1
1
1
2
4



53

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



54

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



55

LMD - Partitionnement

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

COUNT (VOLNUM)
Par partition
Pour Plnum=1
COUNT(Volnum)=5

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

56

LMD - Partitionnement

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

Plnum	COUNT(Volnum)
1	5
2	1
4	1



57

LMD - Partitionnement

```
SELECT Pnum,
       COUNT (Volnum) AS "Nombre de vols"
FROM VOL
GROUP BY Pnum ;
```

Pnum	Nombre de vols
1	5
2	1
4	1



58

LMD - Partitionnement

- Nombre de vols par pilote et par avion ?

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

Partition
par Pnum

Volnum	Pnum	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



59

LMD - Partitionnement

- Nombre de vols par pilote et par avion ?

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

Partition
par Pnum et
par Avnum

Volnum	Pnum	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



60

LMD - Partitionnement

- Nombre de vols par pilote et par avion ?

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

Pnum	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



61

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 Pnom, COUNT (*)
FROM PILOTE, VOL
WHERE PILOTE.Pnum=VOL.Pnum
AND VD = 'PARIS'
GROUP BY Pnom;
```



62

LMD - Partitionnement

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

```
SELECT AVION.Avnum,
       Avnom,
       COUNT (VOL.Pnum)
FROM PILOTE, VOL, AVION
WHERE VOL.Pnum=AVION.Avnum
AND PILOTE.Pnum=VOL.Pnum
GROUP BY AVION.Avnum, Avnom;
```



63

LMD - Partitionnement

Pnum	Plnom	Avnum	Avnom	Volnum	Pnum	Avnum
1	DURAND	10	AIRBUS	V1	1	10
2	DUPOND	20	AIRBUS	V2	2	10

AV 10 : P 1 (V1) AV20 : P 1 (V3)

P 2 (V2) P 2 (V5)

P 2 (V4)

P 1 (V6)

= 4 pilotes = 2 pilotes

(Il n'y a que 2 pilotes dans la base !)

64

LMD - Partitionnement

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

```
SELECT AVION.Avnum,
       Avnom,
       COUNT (DISTINCT VOL.Pnum)
FROM PILOTE, VOL, AVION
WHERE VOL.Pnum=AVION.Avnum
AND PILOTE.Pnum=VOL.Pnum
GROUP BY AVION.Avnum, Avnom;
```

65

LMD - Partitionnement

- Les éléments qui sont dans le GROUP BY doivent se retrouver dans la SELECT

```
SELECT AVION.Avnum,
       Avnom,
FROM ..
GROUP BY AVION.Avnum, Avnom;
```

- En fait ce n'est pas obligatoire d'après la norme mais de très nombreux SGBD ne le supporte pas

66

LMD - Partitionnement

- Les éléments qui sont dans le GROUP BY doivent se retrouver dans la SELECT sauf s'il n'y a qu'une fonction agrégative dans le SELECT

```
SELECT COUNT (*)
FROM VOL
GROUP BY Plnum;
```

Compte le nombre de vols par pilote



67

LMD - Partitionnement

- Condition pour la partition

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



68

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 retourne un résultat que si le nombre de vol pour un pilote est supérieur à 5



69

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



70

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 Avnom LIKE 'AIRBUS%'
```

= RES1



71

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
```



72

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%');
```



73

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**



74

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 Plnum=110)
```

Remarque :
Ajouter dans la requête principale
AND Plnum != 110
pour ne pas avoir le pilote 110

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



75

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.Plnum=Vol.Plnum) ;
```

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

76

LMD – Requêtes corrélées

- Quels sont les pilotes qui effectuent des vols ?

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

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

77

LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?

Plnum	Plnom
1	DUPONT
2	DUPONT
3	DURAND

- <1, 'DUPONT'> et <2, 'DUPONT'> sont homonymes

Résultat attendu :

Plnum	Plnom
1	DUPONT
2	DUPONT

78

LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?

R1 = **JOINTURE**(PILOTE, PILOTE/PInom=PInom)

R2 = **JOINTURE**(PILOTE, PILOTE/PInum=PInum)

R3 = **DIFFERENCE**(R1,R2)

R4 = **PROJECTION**(R3/R1.PInum, R2.PInum)



79

LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?

R1

PInum	PInom	PInum	PInom
1	DUPONT	1	DUPONT
1	DUPONT	2	DUPONT
2	DUPONT	1	DUPONT
2	DUPONT	2	DUPONT
3	DURAND	3	DURAND

R2

PInum	PInom	PInum	PInom
1	DUPONT	1	DUPONT
2	DUPONT	2	DUPONT
3	DURAND	3	DURAND

R4

PInum	PInom
1	DUPONT
2	DUPONT

R3

PInum	PInom	PInum	PInom
1	DUPONT	2	DUPONT
2	DUPONT	1	DUPONT

80



LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?
- Utilisation de la différence

SELECT p1.PInum, p1.PInom, p2.PInum, p2.PInom

FROM PILOTE p1, PILOTE p2

WHERE p1.PInom = p2.PInom

MINUS

SELECT p1.PInum, p1.PInom, p2.PInum, p2.PInom

FROM PILOTE p1, PILOTE p2

WHERE p1.PInum = p2.PInum;

PInum	PInom	PInum	PInom
1	DUPONT	2	DUPONT
2	DUPONT	1	DUPONT

Difficile de ne conserver que les premières colonnes

81



LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?
- Une autre vision de la différence

```
SELECT p1.num, p1.nom
FROM PILOTE p1, PILOTE p2
WHERE p1.num = p2.num
AND (p1.num, p1.nom, p2.num, p2.nom) NOT IN (SELECT
    p3.num, p3.nom, p4.num, p4.nom
    FROM PILOTE p3, PILOTE p4
    WHERE p3.num = p4.num);
```

Attention
Il faut les 4 attributs

Requête principale et
Requête imbriquée
indépendantes

Pnum	Pnom
1	DUPONT
2	DUPONT

82

LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?
- Avec une requête corrélée

```
SELECT *
FROM PILOTE p1
WHERE EXISTS (SELECT *
    FROM PILOTE p2
    WHERE p1.num != p2.num
    AND p1.nom = p2.nom);
```

Pnum	Pnom
1	DUPONT
2	DUPONT

83

LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?
- Avec un partitionnement

```
SELECT Pnum, Pnom
FROM PILOTE
WHERE Pnom IN (SELECT Pnom
    FROM PILOTE
    GROUP BY Pnom
    HAVING COUNT (*) > 1);
```

Pnum	Pnom
1	DUPONT
2	DUPONT

84

LMD – Requêtes corrélées

- Existe t'il des homonymes parmi les pilotes ?
- Avec une condition sur les numéros – Résultats un peu différents

```
SELECT p1.Plnum, p2.Plnum, p1.Plnom
FROM PILOTE p1, PILOTE P2
WHERE p1.Plnom=p2.Plnom
AND p1.Plnum < p2.Plnum;
```

p1.Plnum	p2.Plnum	P1.Plnom
1	2	DUPONT

85

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 :

x1	y1
x1	y2
x2	y1
x2	y3

 \div

y1
y2

 \rightarrow

x1

86

DIVISION - Rappel algébrique

- Avions conduits par tous les pilotes :
VOL1 (PLNUM \div 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

Quotient

Diviseur PIL2

PLNUM
102
103



AVNUM
30
32

Quotient

87

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



88

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
----	----



89

La division en SQL

- Pour avoir les bons :

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

x1	-	x2	=	x1
x2				



90

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



91

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 :



92

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.Plnum=PILOTE.Plnum
                                      AND VOL.Avnum=AVION.Avnum));

```

Bloc 2

Bloc 3



93

La division en SQL

Existe-t-il un pilote tel qu'il n'existe aucun avion de la compagnie qui ne soit pas conduit par ce pilote ?

Un pilote est sélectionné	SELECT *	
	FROM PILOTE	Bloc 1
s'il n'existe aucun	WHERE NOT EXISTS (
avion	SELECT *	
	FROM AVION	Bloc 2
pour lequel il n'y a aucun	WHERE NOT EXISTS (
vol	SELECT *	
	FROM VOL	
(pour ce pilote et cet avion)	WHERE VOL.Plnum=PILOTE.Plnum	
	AND VOL.Avnum=AVION.Avnum));	Bloc 3



94

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

Plnum	...	Avnum	...	Volnum	Plnum	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 donc 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.



La division en SQL

PInum	...	Avnum	...	Volnum	PInum	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.



97

La division en SQL

- Les pilotes conduisant tous les airbus

Bloc 1

SELECT *

FROM PILOTE

WHERE NOT EXISTS (SELECT *

Bloc 2

FROM AVION

WHERE Avnom LIKE 'AIRBUS%'

AND NOT EXISTS (SELECT *

Bloc 3

FROM VOL

WHERE VOL.PInum=PILOTE.PInum

AND VOL.Avnum=AVION.Avnum));



98

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 ? »



99

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);
```



100

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



101

La division en SQL

- Les pilotes conduisant tous les airbus


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

Attention la condition doit être dans les deux



102

- Des questions ?

103
