SQL

Blocs SFW, Sous-requêtes,
Jointures, Groupes et Agrégats

Notes préparées par O. Boucelma

Source : J. Ullman

1

1

Blocs SFW (Select-From-Where)

SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <condition> (sur les tuples des relations)

2

2

Base de données test

Bières(nom, marque)
Bars(nom, adr, licence)
Buveurs(nom, adr, téléphone)
Boit(buveur, bière)
Vends(bar, bière, prix)
Fréquente(buveur, bar)

3

3

Exemple

- Bières(nom, marque)
  - Bières brassées par Anheuser-Busch ?

```
SELECT nom
FROM Bières
WHERE marque = 'Anheuser-Busch';
```

Les valeurs de types string sont entre apostrophes. SQL est *neutre* vis à vis des majuscules/minuscules sauf pour les strings.

4

4

Résultat de la requête

nom
Bud
Bud Lite
Michelob
. . .

5

5

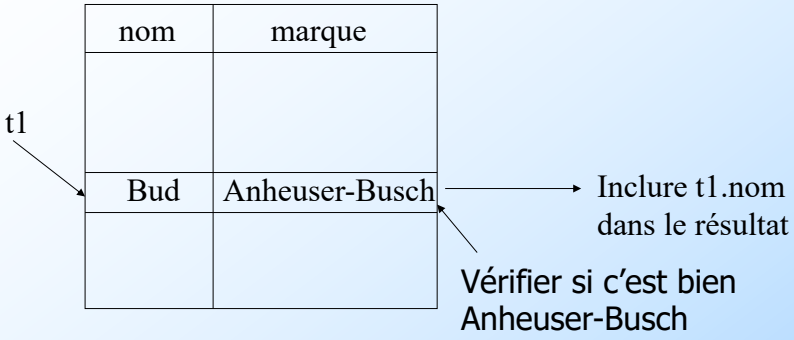
Sens d'une requête mono-relation

- Début : considérer la relation de la clause FROM.
- Appliquer la sélection de la clause WHERE.
- Appliquer la projection indiquée dans la clause SELECT.

6

6

Sémantique opérationnelle



7

7

Sémantique opérationnelle

- Considérons une variable de tuple *t* pour la relation de la clause FROM.
  - La variable *t* va parcourir tous les tuples de la relation.
  - Tester si le tuple pointé par *t* satisfait la clause WHERE.
  - Si c'est le cas, calculer les attributs ou expressions du SELECT en utilisant les composantes de ce tuple.

8

8

Le caractère * dans un SELECT

- Le caractère * est un alias pour tous les attributs de la relation qui se trouvent dans la clause WHERE
- Exemple : Bières(nom, marque)

```
SELECT *
FROM Bières
WHERE marque = 'Anheuser-Busch';
```

9

9

Résultat de la requête

nom	marque
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
. . .	. . .

10

10

Renomage des attributs

- On peut changer les noms d'attributs dans le résultat en utilisant la qualification "AS <nouveau nom>"
- Bières(nom, marque):

```
SELECT nom AS bière, marque
FROM Bières
WHERE marque = 'Anheuser-Busch'
```

11

11

Résultat de la requête

bière	marque
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
. . .	. . .

12

12

Expressions dans les clauses SELECT

- Toute expression (arithmétique ou autre) "qui a un sens" peut faire partie d'une clause SELECT.
- Vends(bar, bière, prix)

```
SELECT bar, bière,
   prix * 6.56 AS prixEnFrancs
FROM Vends;
```

13

13

Résultat de la requête

bar	bière	prixEnFrancs
laMarine	Bud	15,5
laPlaine	Miller	18,2
…	…	…

14

14

Expressions constantes

- Boit(buveur, bière)

```
SELECT buveur,
    'aime la Bud' AS buveurDeBud
FROM Boit '
WHERE bière = 'Bud';
```

15

15

Résultat de la requête

buveur	buveurDeBud
Sophie	aime la Bud
Fred	aime la Bud
…	…

16

16

Conditions complexes dans la clause WHERE

- Vends(bar, bière, prix)
- Prix de la Bud au Joe's Bar

Noter comment on spécifie l'apostrophe de Joe's.

```
SELECT prix
FROM Vends
WHERE bar = 'Joe''s Bar' AND
    bière = 'Bud';
```

17

17

Patterns

- Les clauses WHERE peuvent contenir des conditions dans lesquelles une valeur de type string est comparée à un pattern (modèle) du style :
  - <Attribut> LIKE <pattern> ou bien <Attribut> NOT LIKE <pattern>
  - <pattern> contient % ou _
    % pour n'importe quelle chaîne
    _ pour n'importe quel caractère

18

18

Exemple

Buveurs(nom, adr, téléphone)

Buveurs ayant un numéro de portable :

```
SELECT nom
FROM Buveurs
WHERE téléphone LIKE '06%';
```

19

19

Valeurs NULL

- Des valeurs NULL peuvent apparaître dans un tuple. Leur sens dépend du contexte. On a deux cas principalement :
  - *Valeur manquante* : par exemple, on sait que le bar laMarine a une adresse mais on ne la connaît pas.
  - *Valeur inapplicable* : la valeur de l'attribut épouse (ou époux) pour une personne célibataire .

20

20

Comparaison des NULL

- Logique à 3 valeurs dans SQL :
  - TRUE, FALSE, UNKNOWN.
- Une valeur comparée à NULL, retourne un résultat UNKNOWN.
- Une requête produit un tuple si la condition du WHERE est TRUE.

21

21

Logique à 3 valeurs : la 3-logique

- Considérons les opérateurs AND, OR, et NOT dans une 3-logique
  - **Conventions** :
    TRUE = 1, FALSE = 0, et UNKNOWN = ½.
    AND = MIN; OR = MAX, NOT(x) = 1-x.
- Exemple de calcul
  - TRUE AND (FALSE OR NOT(UNKNOWN)) =
    MIN(1, MAX(0, (1 - ½))) =
    MIN(1, MAX(0, ½) = MIN(1, ½) = ½.

22

22

Exemple

- Considérons le cas suivant

bar	bière	prix
laMarine	Bud	NULL

```
SELECT bar
FROM Vends
WHERE prix < 12.5 OR prix >= 12.5;
```

UNKNOWN UNKNOWN

UNKNOWN

23

23

2-logique != 3-logique

- Des lois comme la commutativité du AND, sont valables dans la 3-logique.
- Mais pas d'autres
  - Exemple : *p* OR NOT *p* = TRUE.
  - Si *p* = UNKNOWN, on a :
    p OR NOT *p* = MAX(½, (1 – ½))
    = ½ (!= 1)

24

24

Requêtes avec plusieurs relations

25

25

Requêtes avec plusieurs relations

- En pratique, une requête implique plusieurs relations.
- Ces relations sont spécifiées dans la clause FROM.
- Les attributs homonymes sont distinguables par leur nom "<relation>.<attribut>"

26

26

Exemple

- Boit(buveur, bière), Fréquente(buveur, bar)
  - Bières bues par (au moins) une personne qui fréquente le bar laMarine.

```
SELECT bière
FROM Boit, Fréquente
WHERE bar = 'laMarine' AND
   Fréquente.buveur =
   Boit.buveur;
```

27

27

Sens d'une requête

1. Considérer le produit de toutes les relations de la clause FROM.
2. Appliquer la condition de sélection de la clause WHERE.
3. Projeter sur la liste des attributs et expressions de la clause SELECT.

28

28

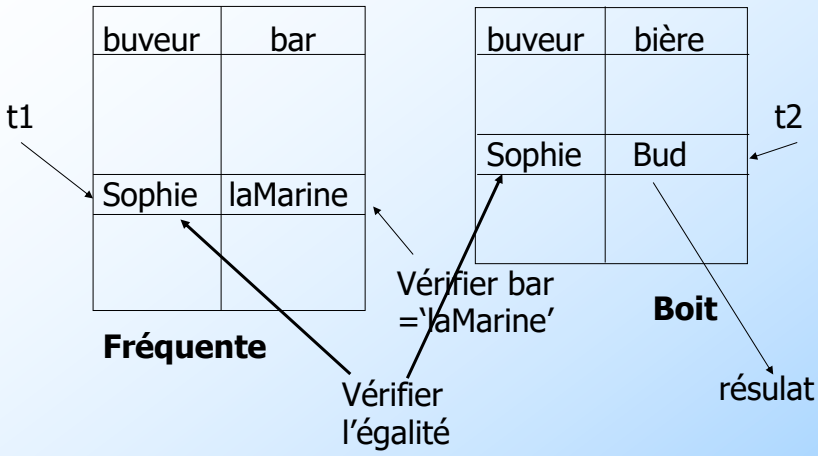
Sémantique opérationnelle

- Considérons une variable de tuple *t* pour chaque relation de la clause FROM.
  - Les variables *t* vont construire toutes les combinaisons de tuples possibles en parcourant chacune leur relation.
  - Si les variables *t* pointent sur des tuples qui satisfont la clause WHERE, alors retenir ces tuples dans la sélection.

29

29

Exemple



30

30

Variables de tuples explicites

- Une requête peut utiliser deux copies de la même relation.
- Ces copies seront distinguées à l'aide des noms de variables de tuples écrits à la suite des noms de relation dans la clause FROM.
- Pratique courante de renomage (mais pas obligatoire) .

31

31

Exemple

- Donner tous les couples de bières provenant de la même brasserie :
  - En évitant les paires (Bud, Bud).
  - En triant le résultat par ordre alphabétique, c'est-à-dire qu'on retient les couples (Bud, Miller) et non pas (Miller, Bud).

```
SELECT b1.nom, b2.nom
FROM Bières b1, Bières b2
WHERE b1.marque = b2.marque AND
   b1.nom < b2.nom;
```

32

32

Sous-requêtes

- Un bloc SFW mis entre parenthèses (*sous-requête*) peut être utilisé comme valeur dans de nombreux endroits, y compris dans les clauses FROM et WHERE.
  - Exemple : à la place d'une relation d'une clause FROM, on peut mettre une autre requête, et ensuite interroger le résultat de cette requête.
  - On utilise une variable de tuple pour nommer les tuples du résultat.

33

33

Sous-requêtes renvoyant un seul tuple

- Si une sous-requête renvoie un et un seul tuple alors ce tuple peut être utilisé comme valeur
  - En général, le tuple contient un seul attribut
- On a une erreur au run-time si
  - aucune valeur n'est retournée
  - plusieurs valeurs sont retournées

34

34

Exemple

- Vends(bar, bière, prix)
  - Trouver les bars qui servent la Miller au même prix que de la Bud au bar de laMarine.
- Résultat = combinaison de 2 requêtes :
  1. Trouver le prix X de la Bud chez laMarine
  2. Trouver les bars qui vendent la Miller au prix X.

35

35

Solution = Requête + sous-requête

```
SELECT bar
FROM Vends
WHERE bière = 'Miller' AND
    prix = (SELECT prix
      FROM Vends
      WHERE bar = 'laMarine'
        AND bière = 'Bud');
```

Le prix de la Bud chez laMarine

36

36

L'opérateur IN

- <tuple> IN <relation> est VRAI si le tuple fait partie de la relation.
  - <tuple> NOT IN <relation> est l'inverse
- Une <expression> IN peut apparaître dans une clause WHERE.
- <relation> est souvent une sous-requête.

37

37

Exemple

- Bières(nom, marque) et Boit(buveur, bière)
  - Trouver le nom et la marque des bières bues par Fred.

```
SELECT *
FROM Bières
WHERE nom IN (SELECT bière
              FROM Boit
              WHERE buveur = 'Fred');
```

Bières bues par Fred

38

38

L'opérateur Exists

- EXISTS(<relation>) est VRAI si et ssi la <relation> n'est pas vide.
- Exemple : Bières(nom, marque)
  - Trouver les bières *b* dont le fabricant ne fabrique exactement qu'une seule marque de bière (celle de *b*).

39

39

Exemple

```
SELECT nom
FROM Bières b1
WHERE NOT EXISTS (
        SELECT *
        FROM Bières
        WHERE marque = b1.marque AND
              nom <> b1.nom);
```

Portée des noms : marque réfère à la relation Bières qui est dans le bloc le plus proche

Bières de la même marque que b1 mais pas b1.

40

40

L'opérateur ANY

- *x* = ANY(<relation>) est une condition booléenne qui vaut VRAI si *x* est égal au moins à un tuple de la relation.
- L'opérateur = peut être remplacé par d'autres opérateurs de comparaison.
  - Exemple: *x* >= ANY(<relation>) signifie que *x* n'est pas le plus petit tuple de la relation.
  - les tuples doivent avoir une seule composante.

41

41

L'opérateur ALL

- *x* <> ALL(<relation>) est une condition booléenne qui vaut VRAI si et ssi pour tout tuple *t* de la relation, *x* n'est pas égal à *t*.
  - Donc *x* ne fait pas partie de la relation.
- L'opérateur <> peut être remplacé par d'autres opérateurs de comparaison
  - Exemple : *x* >= ALL(<relation>)
    il n'existe pas de tuple supérieur à *x* dans la relation.

42

42

Exemple

- Vends(bar, bière, prix)
  - Bières vendues au prix le plus élevé

```
SELECT bière
FROM Vends
WHERE prix >= ALL(
      SELECT prix
      FROM Vends);
```

Le prix ne doit pas être inférieur à tout prix du second bloc SELECT FROM.

43

43

Union, Intersection, et Différence

- Union, intersection, et différence de relations s'expriment ainsi :
  - (sous-requête) UNION (sous-requête)
  - (sous-requête) INTERSECT (sous-requête)
  - (sous-requête) EXCEPT (sous-requête)

44

44

Exemple

- Boit(buveur, bière), Vends(bar, bière, prix), et Fréquente(buveur, bar)
- Trouver les buveurs et les bières vérifiant :
  1. le buveur boit une bière, et
  2. le buveur fréquente au moins un bar qui vend cette bière.

45

45

Solution

Le buveur Fréquente un bar qui vend la bière en question.

```
(SELECT * FROM Boit)
  INTERSECT
(SELECT buveur, bière
 FROM Vends, Fréquente
 WHERE Fréquente.bar = Vends.bar
);
```

46

46

Sémantique ensembliste ?

- Un bloc SFW utilise une sémantique de multi-ensemble (bag) alors que les opérateurs union, intersection, et différence utilisent une sémantique ensembliste (set).
  - {1,2,3} est un ensemble (pas de doublon)
  - {1,2,2,3} est un bag

47

47

Motivation : efficacité

- En projetant des tuples, il est plus aisé (efficace) d'éviter l'élimination des doublons.
  - Traiter les tuples à la volée (*one tuple-at-a-time*)
- Pour l'intersection ou la différence, c'est plus efficace de trier les relations d'abord.
  - Dans ce cas autant supprimer les doublons.

48

48

Gestion des doublons

- On peut transformer un bag en un set :
  - SELECT DISTINCT . . .
- On peut forcer le résultat à être un bag en utilisant la clause ALL
  - ... UNION ALL . . .

49

49

Exemple : DISTINCT

- Vends(bar, bière, prix)
  - Liste des prix des bières

```
SELECT DISTINCT prix
FROM Vends;
```

- Sans la clause DISTINCT, chaque prix aurait été affiché autant de fois qu'il y a un couple (bar,bière) correspondant à ce prix.

50

50

Exemple : ALL

- Liste des buveurs qui fréquentent plus de bars qu'ils ne boivent de bières.
  - Fréquente(buveur, bar), Boit(buveur, bière)

```
(SELECT buveur FROM Fréquente)
    EXCEPT ALL
(SELECT buveur FROM Boit);
```

  - Dans le résultat, on peut avoir des doublons
  - Tous les résultats sont comptabilisés, autant de fois que nécessaire

51

51

Jointures

- Plusieurs expressions de jointure (de bag) dans SQL
- Expressions requêtes seules ou à la place de relations dans une clause FROM

52

52

Produits et Jointures naturelles

- Jointure naturelle :

  R NATURAL JOIN S;
- Produit :

  R CROSS JOIN S;
- Exemple:

```
Boit NATURAL JOIN Serves;
```
- Les relations peuvent être des sous-requêtes parenthésées.

53

53

Théta Jointure

- R JOIN S ON <condition>
  - Buveurs(nom, adr), Fréquente(buveur, bar)

```
Buveurs JOIN Fréquente ON
    nom = buveur;
```

Résultat : quadruplets (*n, a, n, b*) tels que le buveur de nom *n* vit à l'adresse *a* et fréquente le bar *b*.

54

54

Jointures externes

- R OUTER JOIN S est l'expression de base de la jointure externe. Les variantes sont obtenues en mettant de façon optionnelle :
  1. NATURAL après OUTER.
  2. ON <condition> après JOIN.
  3. LEFT, RIGHT, ou FULL (par défaut) avant OUTER.
     - LEFT = "NULLiser" dans S les tuples non appariés de R.
     - RIGHT = "NULLiser" dans R les tuples non appariés de S.
     - FULL = "NULLiser" les tuples non appariés de R et S.

55

55

Motivation de la jointure externe

- Dans une opération R JOIN S normale, on ne garde que les tuples appariés
  - Donc on perd l'information sur les autres tuples (qui ne *matchent* pas)
  - LEFT : on prend les tuples non appariés de R et on complète avec des NULL sur les autres attributs de S pour construire le résultat final.

56

56

Exemple de jointure externe

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

(1,2) est joint avec (2,3), mais les 2 autres tuples ne matchent pas

R OUTER JOIN S =

A	B	C
1	2	3
4	5	NULL
NULL	6	7

57

57

Agrégations

- SUM, AVG, COUNT, MIN, et MAX
- S'appliquent à une colonne de relation pour produire l'agrégat sur cette colonne.
- COUNT(*) compte le nombre de tuples.

58

58

Exemple d'agrégation

- Vends(bar, bière, prix)
  - Prix moyen de la Bud :

```
SELECT AVG(prix)
FROM Vends
WHERE bière = 'Bud';
```

59

59

Elimination des doublons

- On utilise la clause DISTINCT à l'intérieur de l'agrégation.
- Exemple : nombre des prix différents pour la bière Bud

```
SELECT COUNT(DISTINCT prix)
FROM Vends
WHERE bière = 'Bud';
```

60

60

Agrégation et valeurs nulles

- Une valeur NULL ne participe pas au résultat d'un opérateur d'agrégation
- Mais si toutes les valeurs d'une colonne sont NULL alors le résultat de l'agrégation est NULL.

Effet des valeurs NULL

```
SELECT count(*)
FROM Vends
WHERE bière = 'Bud';
```

Nombre de bars qui vendent la Bud.

```
SELECT count(prix)
FROM Vends
WHERE bière = 'Bud';
```

Nombre de bars qui vendent la Bud à un prix connu.

Groupement

- Expression SFW suivie par l'opérateur GROUP BY et une liste d'attributs.
- Le résultat du SFW est groupé selon les valeurs des attributs. Toute opération d'agrégation est effectuée uniquement sur chacun des groupes.

63

63

Exemple de GROUP BY (1)

- Vends(bar, bière, prix)
- Prix moyen de chaque bière

```
SELECT bière, AVG(prix)
FROM Vends
GROUP BY bière;
```

64

64

Exemple de GROUP BY (2)

- Vends(bar, bière, prix) et Fréquente(buveur, bar)
- Pour chaque buveur donner le prix moyen de la Bud dans les bars qu'ils fréquentent

```
SELECT buveur, AVG(prix)
FROM Fréquente, Vends
WHERE bière = 'Bud' AND
      Fréquente.bar = Vends.bar
GROUP BY buveur;
```

Calculer les triplets buveur-bar-prix pour la Bud d'abord, ensuite grouper par buveur.

65

65

Restriction sur les attributs du SELECT avec l'agrégation

- Si un opérateur d'agrégation est utilisé, chaque élément de la liste SELECT doit être soit :
  1. Agrégé, ou
  2. Un attribut dans la liste GROUP BY.

66

66

Requête illicite

```
SELECT bar, MIN(prix)
FROM Vends
WHERE bière = 'Bud';
```

- Réponse = la bière la moins chère ?
- Non car cette requête est illicite
  - Car "on ne sait pas" sur quoi porte le MIN

67

67

Clause HAVING

- HAVING <condition> peut se placer juste après une clause GROUP BY
- La condition s'applique à chaque groupe.
- Les groupes qui ne vérifient pas la condition sont éliminés.

68

68

Exemple : HAVING

- Vends(bar, bière, prix) et Bières(nom, marque)
- Quel est le prix des bières qui sont servies dans au moins 3 bars ou bien qui sont fabriquées par Heineken.

69

69

Solution

Groupes de bières qui ont au moins 3 valeurs de bar non NULL et groupes de bières brassées par Heineken

```
SELECT bière, AVG(prix)
FROM Vends
GROUP BY bière
HAVING COUNT(bar) >= 3 OR
bière IN (SELECT nom
          FROM Bières
          WHERE marque = 'Heineken');
```

Bières brassées par Heineken

70

70

Contraintes des conditions de la clause HAVING

- Les conditions font référence à toute relation ou variable de tuple dans la clause FROM
- Elles peuvent faire référence à un attribut dans la mesure où l'attribut a un sens dans le groupe i.e., c'est soit :
  1. L'attribut de regroupement ou
  2. L'attribut agrégé

71

71