

DataLog

Règles logiques
 Récursion

1

Un langage logique de requêtes (ProLog restreint)

- ◆ Si-alors(-sinon) sont utilisées couramment
 - ▶ Aujourd'hui: mémoire d'entreprise
- ◆ Règles non récursives
 = algèbre relationnelle
- ◆ Règles récursives:
 extension ajoutée à SQL 99

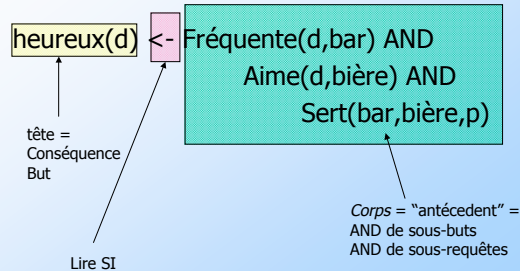
2

Règles logiques

- ◆ Premier exemple
 - ▶ Fréquente(client,bar),
 - ▶ Aime(client,bière),
 - ▶ Sert(bar,bière,prix).
- ◆ Clients "heureux"
 ceux qui fréquentent un bar
 qui sert une bière qu'ils aiment

3

Structure d'une règle / clause



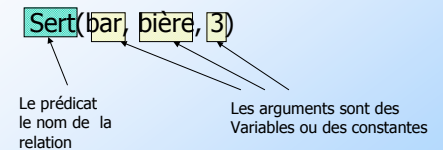
4

Sous-requêtes

- ◆ Un *atome* est un *prédicat*, ou relation avec des variables ou des constantes comme arguments.
- ◆ La tête est un atome;
 le corps est la conjonction

5

Exemple d'atome



6

Interprétation des Règles

Sens d'une clause:

La tête est vraie s'il existe des valeurs des variables qui rendent vraies les atomes du corps de la clause.

7

exemple: Interpretation

`heureux(d) <- Fréquente(d,bar) AND
 Aime(d,bière) AND Sert(bar,bière,p)`

d est heureux
 S'il existe un bar et une bière et un prix
 Tels que
 - *d fréquente bar*
 - *d aime bière*
 - *bar sert bière* au prix *p*

8

Sous requêtes conditionnelles

(atomes de comparaison
 utilisant des prédicats prédéfinis)

- ◆ En plus des atomes relationnels.
 On peut avoir des conditions usuelles:
 par exemple: $x < y$ ou $x < > y$ ou $x = y$.

9

Exemple de conditions

Une bière est bon marché si au moins deux bars la servent à moins de 2€50

Bon_marché(bière)

```
<- Sert(bar1,bière,p1) AND
   Sert(bar2,bière,p2) AND
   p1 < 2,5 AND
   p2 < 2,5 AND
   bar1 <> bar2
```

10

Sous requêtes négatives

- ◆ Une sous requête peut être précédée de NOT
 - ◆ Exemple: Arc(a,b) arc de a à b dans un graphe.
 - ◆ S(x,y) il y a un chemin de longueur 2 de x à y mais pas d'arc direct (contredit la transitivité)
- S(x,y) <- Arc(x,z) AND Arc(z,y)
AND NOT Arc(x,y)

11

Clauses correctes ou sûres

- ◆ Une clause est correcte si:
Toute variable apparaît dans une sous requête non niée,
- ◆ On n'utilise que des clauses correctes
- ◆ Pourquoi? TSVP

12

Exemples de clauses incorrectes

- ◆ Exemples de clauses incorrectes:
 1. S(x) <- NOT R(x)
 2. S(x) <- R(y) AND NOT R(x)
 3. S(x) <- R(y) AND x > y
- ◆ A chaque fois on peut avoir une infinité de valeurs pour "x" même si R est une relation finie.

13

Evaluation des règles

- ◆ Deux approches:
 1. *Instanciation des Variables* : essayer toutes les valeurs possibles pour les variables des sous requêtes, et si les sous requêtes sont validées ajouter le n-uplet
 2. *Instanciation des n-uplets* : Essayer toutes les n-uplets possibles des atomes sans négation, et si les sous requêtes sont validées ajouter le n-uplet.

14

Exemple: Variables --- 1

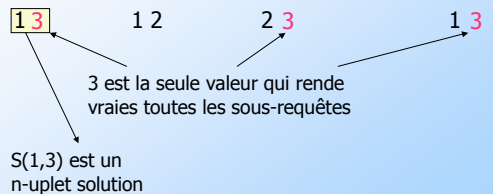
S(x,y) <- Arc(x,z) AND Arc(z,y)
AND NOT Arc(x,y)

- ◆ Arc: Arc(1,2) et Arc(2,3)
- ◆ Deux instaciations rendent Arc(x,z) vrai:
 1. x = 1; z = 2
 2. x = 2; z = 3

15

Exemple: Variables x=1, z=2

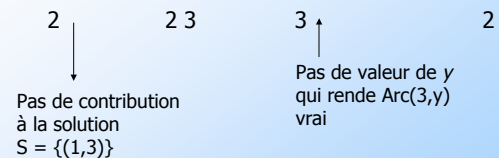
S(x,y) <- Arc(x,z) AND Arc(z,y) AND NOT Arc(x,y)



16

Exemple: Variables x=2, z=3

S(x,y) <- Arc(x,z) AND Arc(z,y) AND NOT Arc(x,y)



17

Instanciation des n-uplets

- ◆ Commencer par les atomes relationnels non niés.
- ◆ Considérer toutes les instanciations de leur n-uplets par des n-uplets pris dans les relations correspondantes.
- ◆ Si ces instanciations donnent des valeurs cohérentes aux variables et rendent toutes les sous-requêtes vraies alors ajouter à la solution le n-uplet correspondant.

18

Exemple: avec n-uplets

$S(x,y) \leftarrow \text{Arc}(x,z) \text{ AND } \text{Arc}(z,y) \text{ AND NOT } \text{Arc}(x,y)$
 $\text{Arc}(1,2), \text{Arc}(2,3)$

- ◆ Quatre instanciations possibles pour les deux premiers atomes:

Arc(x,z)	Arc(z,y)
(1,2)	(1,2)
(1,2)	(2,3)
(2,3)	(1,2)
(2,3)	(2,3)

Seule instanciati
 on possible avec
 une valeur
 cohérente pour 'z'.
 Elle valide aussi
 NOT Arc(x,y) donc
 On ajoute S(1,3)
 au résultat.

19

Programmes Datalog (requêtes)

- ◆ Un programme *Datalog* est une suite de clauses
- ◆ Deux sortes de prédicats
 1. EDB = *Extensional Database* = table stockée.
 2. IDB = *Intensional Database* = relation définie par les clauses.
- ◆ Une relation n'est jamais EDB et IDB!
- ◆ Jamais de EDB dans les têtes.

20

Algèbre relationnelle en DataLog (au tableau)

- ◆ Sélection
- ◆ Projection
- ◆ Opérations ensemblistes
- ◆ Produit cartésien
- ◆ Jointure

21

DataLog en algèbre relationnelle

- ◆ Corps de la clause produit, jointure, sélection, différence pour la négation
- ◆ Tête de la clause projection
- ◆ Clauses multiples pour un IDB union

22

Evaluation des programmes Datalog

- ◆ Pas de récursion = ordonner les clauses pour que le corps ne contienne que des prédicats déjà définis et évalués.
- ◆ Si un prédicat IDB est défini par plus d'une clause, chaque clause ajoute des n-uplets au prédicat IDB.

23

Exemple: Programme DataLog

- ◆ En utilisant la table EDB
 $\text{Sert}(\text{bar}, \text{bière}, \text{prix})$
 $\text{Bières}(\text{nom}, \text{braserie})$
- ◆ Trouver les Brasseries produisant des bières que le Lucifer ne sert pas:
 - ◆ $\text{Lucifer}(b) \leftarrow \text{Sert}(\text{'Lucifer'}, b, p)$
 - ◆ $\text{Réponse}(m) \leftarrow \text{bières}(b,m) \text{ AND NOT Lucifer}(b)$

24

Expressivité de Datalog

- ◆ Sans récursion, Datalog équivaut à l'algèbre relationnelle.
- ◆ Avec recursion, Datalog sort de ce cadre
- ◆ Néanmoins pas toutes les opérations calculables sur les relations

25

Exemple récursif dans un graphe orienté acyclique

- ◆ EDB: $\text{Par}(c,p)$:
 p est un des sommets immédiatement au dessus de c .
- ◆ Cousins sommets avec un ancêtre commun :
 - ◆ $\text{FS}(x,y) \leftarrow \text{Par}(x,p) \text{ AND } \text{Par}(y,p) \text{ AND } x < y$
 - ◆ $\text{Cousin}(x,y) \leftarrow \text{FS}(x,y)$
 - ◆ $\text{Cousin}(x,y) \leftarrow \text{Par}(x,xp) \text{ AND } \text{Par}(y,yp) \text{ AND } \text{Cousin}(xp,yp)$

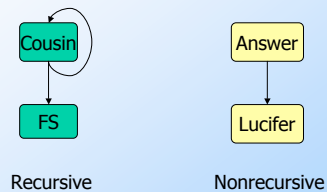
26

Définition de la Récursion

- ◆ Former un graphe de *dépendance*
 $\text{sommets} = \text{prédicat IDB}$
- ◆ Arc $X \rightarrow Y$ ssi X est la tête d'une clause avec Y dans le corps de cette clause
- ◆ Cycle = récursion
- ◆ Pas de cycle = pas de récursion.

27

Exemple: graphe de Dépendance



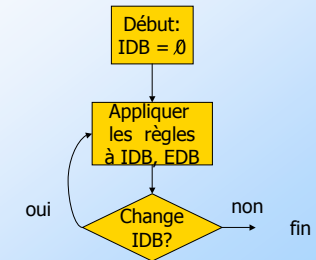
28

Evaluation des programmes récurrents

- ◆ S'il n'y a pas de négation:
 1. Supposer les IDB vides.
 2. Evaluer les clauses en utilisant les EDB et les n-uplets déjà ajoutés aux IDB.
 3. S'arrêter quand le contenu du prédicat IDB est stationnaire.

29

Evaluation Naïve



30

Exemple: Evaluation de Cousin

- ◆ On boucle pour construire des n-uplets de
 - ▶ FS (en rouge)
 - ▶ Cousin (en vert)
- ◆ Rappelons les clauses:

```
FS(x,y) <- Par(x,p) AND Par(y,p) AND x<>y
Cousin(x,y) <- FS(x,y)
Cousin(x,y) <- Par(x,yp) AND Par(y,yp)
AND Cousin(xp,yp)
```

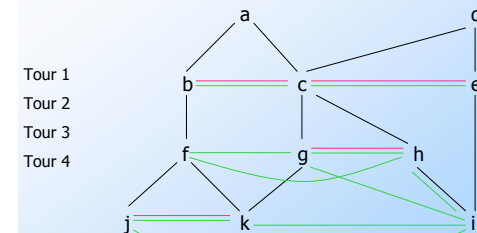
31

Evaluation Semi-Naïve

- ◆ Les EDB ne changent pas, à chaque tour on obtient un nouvel n-uplet dans un IDB que si on utilise un nouvel n-uplet d'un IDB
- ◆ Evitons de redécouvrir les mêmes n-uplets.
 - ▶ Malgré cela un même n-uplet peut être redécouvert.

32

EDB: est un parent de



33

Récursion Plus Négation

- ◆ L'évaluation "Naïve" ne fonctionne plus si des sous-requêtes sont niées.
- ◆ En général, la négation au sein d'un programme récurrent n'a pas de sens.
- ◆ Même quand récursions et négations sont indépendantes, les prédicats IDB peuvent être mal définis.

34

Négation Stratifiée

- ◆ La stratification est une contrainte habituellement requise sur Datalog avec négation et récursion.
- ◆ La négation ne peut plus être imbriquée dans la récursion.
- ◆ On obtient les relations souhaitées.

35

Problème de la Negation Récursive

$P(x) <- Q(x) \text{ AND NOT } P(x)$
EDB: $Q(1), Q(2)$

initial: $P = \{ \}$
tour 1: $P = \{(1), (2)\}$
tour 2: $P = \{ \}$
tour 3: $P = \{(1), (2)\}$,
etc., etc. ...

36

Strates

- ♦ Intuitivement, la *strate* d'un prédicat IDB est le nombre maximal de négations que l'on peut rencontrer durant son évaluation.
- ♦ Négation stratifiée = "strate finie."
- ♦ Dans $P(x) \leftarrow Q(x) \text{ AND NOT } P(x)$, on peut nier P une infinité de fois pour calculer P .

37

Graphe des Strates

- ♦ Formalisation des strates par un graphe:
 - ▶ Sommets = prédicats IDB.
 - ▶ Arcs $A \rightarrow B$ ssi le prédicat A dépend du prédicat B . (Clause $A \leftarrow \dots B \dots$)
 - ▶ Etiquette "-" sur l'arc $A \rightarrow B$ ssi B est nié dans la clause $A \leftarrow \dots B \dots$

38

Négation stratifiée

- ♦ La strate d'un sommet (prédicat IDB) est le maximum d'arcs "-" sur un chemin partant de ce sommet.
- ♦ Un programme Datalog est *stratifié* si tous ces prédicats IDB ont une strate finie: il n'y a pas de cycle avec un "-".

39

exemple

$P(x) \leftarrow Q(x) \text{ AND NOT } P(x)$



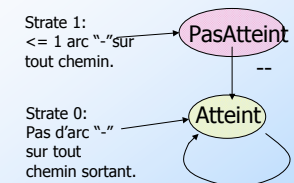
40

Autre exemple

- ♦ EDB = Origine(x), Destination(x), Arc(x,y).
- ♦ Clauses pour "Destinations jamais Atteintes de quelque Origine que ce soit":
 - $\text{Atteint}(x) \leftarrow \text{Origine}(x)$
 - $\text{Atteint}(x) \leftarrow \text{Atteint}(y) \text{ AND Arc}(y,x)$
 - $\text{PasAtteint}(x) \leftarrow \text{Destination}(x) \text{ AND NOT } \text{Atteint}(x)$

41

Graphe des Strates



42

Modèles

- ♦ Un *modèle* est un choix de relations IDB qui avec les relations EDB données rend vraies toutes les clauses, quelles que soient les valeurs données aux variables.
 - ▶ Attention: si le corps de la clause est faux la clause est vraie.
 - ▶ Si le corps de la clause est vrai, la tête doit l'être aussi.

43

Modèles minimaux

- ♦ Pas de négation: un programme Datalog a un unique modèle minimal.
- ♦ Avec la négation, il peut y en avoir plusieurs.
- ♦ Le modèle calculé avec la stratification est celui qui a un sens.

44

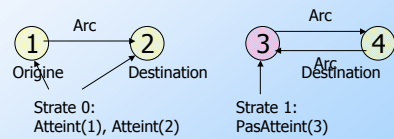
Le modèle stratifié

- ♦ On évalue les prédicats IDB par ordre de strate croissante.
- ♦ Dès qu'un prédicat IDB est évalué, on le considère comme un prédicat EDB.

45

Exemple: Modèles Multiples - 1

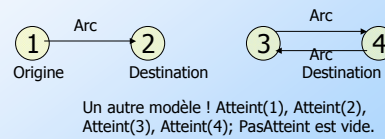
Atteint(x) <- Origine(x)
Atteint(x) <- Atteint(y) AND Arc(y,x)
PasAtteint(x) <- Destination(x) AND NOT Atteint(x)



46

Exemple: Modèles Multiples - 2

Atteint(x) <- Origine(x)
Atteint(x) <- Atteint(y) AND Arc(y,x)
PasAtteint(x) <- Destination(x) AND NOT Atteint(x)



47