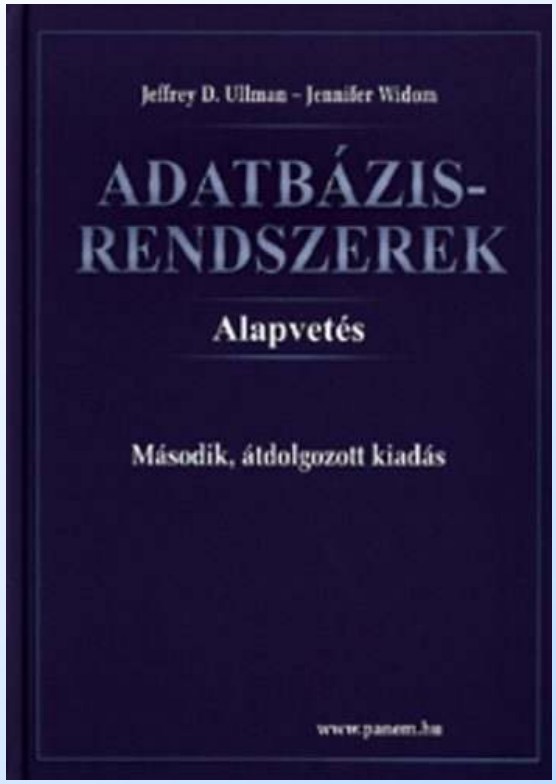
Datalog -1: Logika a relációkhoz



Ullman-Widom: Adatbázisrendszerek

Alapvetés

Második, átdolgozott kiadás, Panem, 2009

5.3. Logika a relációkhoz

5.4. A relációs algebra és Datalog

(Jeffrey D. Ullman, 2007)

Logikai alapú lekérdező nyelv rekurzív lekérdezések

- Predikátum kalkulus
- **E.F. Codd** 1970-ban publikált cikkében def.

A Relational Model of Data for Large Shared Data Banks

- Szabály alapú, Prolog alapján: Datalog

Intuitív bevezetés

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) táblában repülőjáratok adatait tároljuk.

  Mely (x,y) párokra lehet eljutni x városból y városba?

- Datalogban felírva

  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Vagy másképp felírva Datalogban (mi a különbség?)

  Eljut(x, y) <- Jaratok(_, x, y, _, _, _)
  Eljut(x, y) <- Eljut(x, z) AND Eljut(z, y)

Logika ismétlés: ítéletkalkulus

- ítéletváltozók:
  - x, y, z, …
  - igaz/hamis értékek
- kifejezések
  - konstans {I, H}
  - változó
  - ha e, e1, e2 kifejezés, akkor e1 AND e2, e1 OR e2, NOT e, és (e) is kifejezés

Logic As a Query Language

- If-then logical rules have been used in many systems.
- Nonrecursive rules are equivalent to the core relational algebra.
- Recursive rules extend relational algebra and appear in SQL-99.

A Logical Rule

- Our first example of a rule uses the relations Frequents(drinker, bar), Likes(drinker, beer), and Sells(bar, beer, price).
- The rule is a query asking for "happy" drinkers --- those that frequent a bar that serves a beer that they like.

Anatomy of a Rule

Happy(d) <- Frequents(d,bar) AND Likes(d,beer) AND Sells(bar,beer,p)

Head = consequent, a single subgoal

Read this symbol "if"

Body = antecedent = AND of *subgoals*.

Subgoals Are Atoms

- An *atom* is a *predicate*, or relation name with variables or constants as arguments.
- The head is an atom; the body is the AND of one or more atoms.
- Convention: Predicates begin with a capital, variables begin with lower-case.

Example: Atom

Sells(bar, beer, p)

The predicate = name of a relation

Arguments are variables (or constants).

Interpreting Rules

- A variable appearing in the head is *distinguished* ; otherwise it is *nondistinguished*.
- Rule meaning: The head is true for given values of the distinguished variables if there exist values of the nondistinguished variables that make all subgoals of the body true.

Example: Interpretation

Happy(d) <- Frequents(d,bar) AND
Likes(d,beer) AND Sells(bar,beer,p)

Distinguished variable

Nondistinguished variables

Interpretation: drinker *d* is happy if there exist a bar, a beer, and a price *p* such that *d* frequents the bar, likes the beer, and the bar sells the beer at price *p*.

Applying a Rule

- Approach 1: consider all combinations of values of the variables.
- If all subgoals are true, then evaluate the head.
- The resulting head is a tuple in the result.

Example: Rule Evaluation

Happy(d) <- Frequents(d,bar) AND
Likes(d,beer) AND Sells(bar,beer,p)

FOR (each d, bar, beer, p)

IF (Frequents(d,bar), Likes(d,beer), and Sells(bar,beer,p) are all true)

add Happy(d) to the result

- Note: set semantics so add only once.

A Glitch (Fixed Later)

- Relations are finite sets.
- We want rule evaluations to be finite and lead to finite results.
- "Unsafe" rules like P(x)<-Q(y) have infinite results, even if *Q* is finite.
- Even P(x)<-Q(x) requires examining an infinity of *x*-values.

Applying a Rule – (2)

- Approach 2: For each subgoal, consider all tuples that make the subgoal true.
- If a selection of tuples define a single value for each variable, then add the head to the result.
- Leads to finite search for P(x)<-Q(x), but P(x)<-Q(y) is problematic.

Example: Rule Evaluation – (2)

Happy(d) <- Frequents(d,bar) AND
Likes(d,beer) AND Sells(bar,beer,p)

```
FOR (each f in Frequents, i in Likes, and
  s in Sells)
  IF (f[1]=i[1] and f[2]=s[1] and
        i[2]=s[2])
     add Happy(f[1]) to the result
```

Arithmetic Subgoals

- In addition to relations as predicates, a predicate for a subgoal of the body can be an arithmetic comparison.
- We write arithmetic subgoals in the usual way, e.g., $x < y$.

Example: Arithmetic

- A beer is "cheap" if there are at least two bars that sell it for under $2.

```
Cheap(beer) <- Sells(bar1,beer,p1) AND
    Sells(bar2,beer,p2) AND p1 < 2.00
    AND p2 < 2.00 AND bar1 <> bar2
```

Negated Subgoals

- NOT in front of a subgoal negates its meaning.
- Example: Think of Arc(a,b) as arcs in a graph.
  - S(x,y) says the graph is not transitive from *x* to *y*; i.e., there is a path of length 2 from *x* to *y*, but no arc from *x* to *y*.

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

Biztonságos szabályok
Safe Rules

- A rule is *safe* if:
  1. Each distinguished variable,
  2. Each variable in an arithmetic subgoal, and
  3. Each variable in a negated subgoal,

  also appears in a nonnegated, relational subgoal.
- Safe rules prevent infinite results.

Example: Unsafe Rules

- Each of the following is unsafe and not allowed:
  1. S(x) <- R(y)
  2. S(x) <- R(y) AND x < y
  3. S(x) <- R(y) AND NOT R(x)
- In each case, an infinity of *x*'s can satisfy the rule, even if *R* is a finite relation.

An Advantage of Safe Rules

- We can use "approach 2" to evaluation, where we select tuples from only the nonnegated, relational subgoals.
- The head, negated relational subgoals, and arithmetic subgoals thus have all their variables defined and can be evaluated.

Datalog Programs

- *Datalog program* = collection of rules.
- In a program, predicates can be either
  1. EDB = *Extensional Database* = stored table.
  2. IDB = *Intensional Database* = relation defined by rules.
- Never both! No EDB in heads.

Evaluating Datalog Programs

- As long as there is no recursion, we can pick an order to evaluate the IDB predicates, so that all the predicates in the body of its rules have already been evaluated.
- If an IDB predicate has more than one rule, each rule contributes tuples to its relation.

Example: Datalog Program

- Using EDB Sells(bar, beer, price) and Beers(name, manf), find the manufacturers of beers Joe doesn't sell.

JoeSells(b) <- Sells('Joe"s Bar', b, p)

Answer(m) <- Beers(b,m)

AND NOT JoeSells(b)

Example: Evaluation

- Step 1: Examine all Sells tuples with first component 'Joe"s Bar'.
  - Add the second component to JoeSells.
- Step 2: Examine all Beers tuples (b,m).
  - If *b* is not in JoeSells, add *m* to Answer.

Expressive Power of Datalog

- Without recursion, Datalog can express all and only the queries of core relational algebra.
  - The same as SQL select-from-where, without aggregation and grouping.
- But with recursion, Datalog can express more than these languages.

Relációs algebra és Datalog -1

Rel.algebrai műveletek hogyan néznek ki Datalogban?

Halmazműveletek: T.f.h $R(x_1,...,x_n)$, $S(x_1,...,x_n)$ predikátumokhoz tartozó reláció $R(A_1,...,A_n)$, $S(A_1,...,A_n)$

- $R \cap S$ metszetnek megfelelő szabály:

  $Válasz(x_1,...,x_n) \leftarrow R(x_1,...,x_n)$ AND $S(x_1,...,x_n)$

- $R-S$ különbségnek megfelelő szabály:

  $Válasz(x_1,...,x_n) \leftarrow R(x_1,...,x_n)$ AND NOT $S(x_1,...,x_n)$

- $R \cup S$ unió műveletet egyetlen szabállyal nem tudom felírni, mert a törzsben csak AND lehet, OR nem. Ehhez több szabályból álló Datalog program kell:

  Válasz(x1,...,xn)←R(x1,...,xn)
  Válasz(x1,...,xn)←S(x1,...,xn)

Relációs algebra és Datalog -2

Kiválasztás:

- $\sigma_{x_i \theta x_j}(R)$ kifejezésnek megfelelő szabály :
  Válasz$(x_1,\ldots,x_n) \leftarrow R(x_1,\ldots,x_n)$ AND $x_i \theta x_j$
- $\sigma_{x_i \theta c}(E1)$ kifejezésnek megfelelő szabály:
  Válasz$(x_1,\ldots,x_n) \leftarrow R(x_1,\ldots,x_n)$ AND $x_i \theta c$

Vetítés:

- $\Pi_{A_{i1},\ldots,A_{ik}}(R)$ kifejezésnek megfelelő szabály:
  Válasz$(x_{i_1},\ldots,x_{i_k}) \leftarrow R(x_1,\ldots,x_n)$

Megjegyzés: név nélküli anonymus változók, amelyek csak egyszer szerepelnek és mindegy a nevük azt aláhúzás helyettesítheti. Például:

HosszúFilm(c,é) ← Film(c,é,h,_,_,_) AND h ≥ 100

Relációs algebra és Datalog -3

Természetes összekapcsolás: Tegyük fel, hogy $R(A_1,\ldots,A_n, C_1, \ldots, C_k)$ és $S(B_1,\ldots,B_m, C_1, \ldots, C_k)$

- $R \bowtie S$ kifejezésnek megfelelő szabály:

  $Válasz(x_1,\ldots,x_n,y_1,\ldots,y_m, z_1,\ldots, z_k) \leftarrow$

  $\leftarrow R(x_1,\ldots,x_n, z_1, \ldots, z_k)$ AND $S(y_1,\ldots,y_m, z_1, \ldots, z_k)$
- A felírt szabályok biztonságosak.
- Minden Q relációs algebrai kifejezéshez van nem rekurzív, biztonságos, negációt is tartalmazó Datalog program, amelyben egy kitüntetett IDB predikátumhoz tartozó kifejezés ekvivalens a Q lekérdezéssel.
- A nem rekurzív, biztonságos, negációt is tartalmazó Datalog kifejezőerő tekintetében EKVIVALENS a relációs algebrával.