Databázové praktikum

Organizácia kurzu

- Ján Mazák - mazak@dcs.fmph.uniba.sk
- Michal Rjaško – rjasko@dcs.fmph.uniba.sk
- **Hodnotenie:**
  - **3 domáce úlohy po 30 bodov (zadané zhruba v tretinách semestra)**
  - **12 cvičení po 1 bod, treba získať aspoň 9** ---
    do 3 dní po cvičení treba odovzdať e-mailom vyriešené úlohy (aspoň polovicu)

  - A: 90 a viac bodov
  - B: 80 až 89 bodov
  - C: 70 až 79 bodov
  - D: 60 až 69 bodov
  - E: 50 až 59 bodov

Plán kurzu

- Datalog
- SQL, DDL, DML
- práca s databázou v jazyku Java
- Explain - analýza / optimalizácia dotazov
- SQLite, PostgreSQL

Deklaratívne programovanie

- program definuje, **čo** sa má vypočítať, ale nepopisuje, **ako**
- všetky bežné databázové jazyky (o.i. SQL) ani neobsahujú na popis výpočtu prostriedky
- databáza sama volí postup výpočtu podľa dotazu a existujúcich dát

  (netriviálna optimalizácia)

Prolog

- jeden z najbežnejších jazykov pre deklaratívne programovanie
- využíva sa v spracovaní prirodzených jazykov a umelej inteligencii (napr. IBM Watson)
- program pozostáva z faktov a odvodzovacích pravidiel
- interpreter Prologu robí inferenciu (odvodzovanie dôsledkov pravidiel zo známych faktov) pomocou backtrackingu
- využijeme ho ako prostredie na prácu s Datalogom

Prolog / Datalog

- syntax pravidiel:

```
p :- x, y, \+ z.
good_car(X) :- car(X), reliable(X), fast(X), costs_less(X, 30 000).
reliable(toyota).
```

- sémantika:

  p je pravdivé, ak x a y sú pravdivé a z je nepravdivé

- pravidlá definujú nové *predikáty*, napr. good_car, pomocou existujúcich predikátov
- počet argumentov predikátu je *arita*
- sada pravidiel s rovnakou hlavou slúži na vyjadrenie logickej spojky alebo, napr.

```
p :- x.
p :- y.
```

Prolog / Datalog

- príklad datalogovského pravidla:

  res(N, J) :- emp(_, N, J, _, _, S, _, _), S >= 2000.

- na ľavej strane len jeden pozitívny atóm
- premenné začínajú veľkým písmenom
- konštanty malými písmenami
- _ znamená anonymnú premennú

  (ak je _ použité na viac miestach, hodnoty nemusia byť rovnaké, ide o rôzne premenné)
- na vyhodnocovanie aritmetických výrazov slúži operátor is:
  - napr. X is 2+3,
  - nie X = 2+3

    (symbol = je interpretovaný ako unifikácia termov a nedôjde k žiadnej aritmetickej operácii)

Prolog / Datalog

bigger(elephant, horse).

bigger(horse, donkey).

bigger(donkey, dog).

bigger(donkey, monkey).

?- bigger(donkey, dog).

true

?- bigger(monkey, elephant).

false

?- bigger(elephant, monkey).

false

Prolog / Datalog

```
bigger(elephant, horse).
bigger(horse, donkey).
bigger(donkey, dog).
bigger(donkey, monkey).

is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).
```

```
?- is_bigger(monkey, elephant).
false
?- is_bigger(elephant, monkey).
true

?- is_bigger(elephant, X).
X = horse
X = donkey
X = dog
X = monkey
false
```

Negácia

- za pravdivé sú v Prologu pokladané atómy (tvrdenia), ku ktorým je možné odvodiť dôkaz z faktov (pomocou odvodzovacích pravidiel)
- ostatné veci sú nepravdivé (negation as failure)
- nie je možné pridať fakt o nepravdivosti (hlava pravidla neobsahuje negáciu)
- tzv. predpoklad uzavretého sveta (pravdivé je práve to, čo máme v databáze, zvyšok sú nepravdy)
- pravdivosť vo všeobecnosti nie je totožná s dokázateľnosťou

(“táto veta sa nedá dokázať”, viac na matematickej logike)

Negácia

- negácia kombinovaná s rekurziou môže pri interpretácii spôsobiť problémy:

  p :- \+ q.

  q :- \+ p.
- ako vyzerá svet popísaný týmto programom?

  dva stabilné modely: p je pravda a q nie; alebo naopak

  (viac o modeloch na Databázových systémoch)
- všetkým súvisiacim problémom sa vieme vyhnúť, ak budeme používať len *bezpečné pravidlá*: každá premenná sa musí vyskytnúť v nejakom pozitívnom fakte
- príklad nebezpečného pravidla (kvôli Y aj Z):

  p(X, Y) :- a(X), \+ b(Y, Z), Z is Y + 1.

Práca s datalogom: SWI-Prolog

- tri možnosti:
  - na serveri *cvika*, pripojiť sa cez ssh na cvika.dcs.fmph.uniba.sk

    (prihlasovacie meno / heslo ako v AISe)
  - v Linuxe v M217, alebo na vlastnom počítači, kde nainštalujete SWI-Prolog
  - online na https://swish.swi-prolog.org/

- odporúčame otvoriť si 3 okná
  - v jednom editujete súbor s dotazmi, napr. **vim queries_emp.pl**
  - v druhom okne máte spustené prostredie prologu: **swipl -s queries_emp.pl**
  - v treťom okne máte databázu (zoznam faktov)

Práca s datalogom

- po zapísaní dotazu do súboru ho treba uložiť na disk
  (vim: ESC, ":w", ENTER).
- potom novú verziu skompilovať: **make.** (aj s tou bodkou)
  - nezabudnite skontrolovať, či kompilátor hlási chyby a prípadne ich opraviť
- výpočet dotazov:
  ?- q(job(J)).
- predikát "q(_)" slúži na pekné formátovanie výstupu a elimináciu zdanlivých duplikátov (Prolog robí úplný backtracking a konkrétnu hodnotu môže nájsť vo viacerých vetvách)

Databáza EMP

```
%emp(Empno, Ename, Job, Manager, Hiredate, Sal, Deptno)
emp(7839, king, president, null, 19811117, 5000, 10).
emp(7698, blake, manager, 7839, 19810501, 2850, 30).
emp(7782, clark, manager, 7839, 19810609, 1500, 10).
emp(7566, jones, manager, 7839, 19810402, 2975, 20).
emp(7654, martin, salesman, 7698, 19810928, 1250, 30).
emp(7499, allen, salesman, 7698, 19810220, 1600, 30).
emp(7844, turner, salesman, 7698, 19810908, 1500, 30).
emp(7900, james, clerk, 7698, 19811203,  950, 30).
emp(7521, ward, salesman, 7698, 19810222, 1250, 30).
emp(7902, ford, analyst, 7566, 19811203, 3000, 20).
emp(7369, smith, clerk, 7902, 19801217,  800, 20).
emp(7788, scott, analyst, 7566, 19821209, 3000, 20).
emp(7876, adams, clerk, 7788, 19830112, 1100, 20).
emp(7934, miller, clerk, 7782, 19820123, 1300, 10).


%dept(Deptno, Dname, Location)
dept(10, accounting, newyork).
dept(20, research, dallas).
dept(30, sales, chicago).
dept(40, operations, boston).
```